**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»**
Приладобудівний факультет
Кафедра інформаційно-вимірювальних технологій

«До захисту допущено»

Завідувач кафедри ІВТ

_____ Володимир ЄРЕМЕНКО
(підпис) (ініціали, прізвище)

“___”_____2020 р.

# Дипломний проєкт

**на здобуття ступеня бакалавра**

за освітньо-професійною програмою

«Інформаційні вимірювальні технології та системи»

спеціальності 152 «Метрологія та інформаційно-вимірювальна техніка»

на тему: Система тепловізійного прицілу

Виконав (-ла): студент (-ка) IV курсу, групи ВМ-61-2

Шуліков Віталій Юрійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник
доц. каф. ІВС, к.т.н., доц. Самарцев Юрій Миколайович _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент
_____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище ім’я побатькові) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

**Національний технічний університет України**
**«Київський політехнічний інститут**
**імені Ігоря Сікорського»**

Факультет (інститут) *приладобудівний факультет*
(повна назва)

Кафедра *інформаційно-вимірювальних технологій*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 152 «Метрологія та інформаційно-вимірювальна техніка»

Освітньо-професійна програма
«Інформаційні вимірювальні технології та системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Володимир ЄРЕМЕНКО
(підпис) (ініціали, прізвище)

«___»_____20__ р.

**ЗАВДАННЯ**
**на дипломний проєкт студенту**

*Шулікову Віталію Юрійовичу*
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) *Система тепловізійного прицілу*
керівник проєкту (роботи) *Самарцев Юрій Миколайович, к.т.н., доц.* ___ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №___
2. Строк подання студентом проєкту (роботи) _____
3. Вихідні дані до проєкту (роботи) *Система повинна вимірювати відстань до об'єкта з абсолютною похибкою не менше ніж 0,5м. Число вимірювальних каналів – 2.*
4. Зміст пояснювальної записки (перелік завдань, які потрібно розробити) *1. Огляд існуючих рішень. 2. Вибір та обґрунтування схемотехнічного рішення. 3. Розробка структурної схеми системи. 4. Розробка функційної схеми системи. 5. Опис алгоритму роботи системи. 6. Розробка блок-схеми програмного забезпечення системи. 7. Розробка принципової схеми системи. 8.Моделювання роботи системи при наявності балістичного відхилення робочого елемента.*
5. Перелік (ілюстративного) графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів тощо) *Схема структурна. Схема функціональна. Схема принципова.*

6. Консультанти розділів проєкту (роботи)*

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| | | | |
| | | | |

7. Дата видачі завдання *11 лютого 2020 року*

Календарний план

| № з/п | Назва етапів виконання дипломного проєкту (роботи) | Строк виконання етапів проєкту (роботи) | Примітка |
|---|---|---|---|
| 1. | Огляд існуючих рішень | 01.03.2020 р. | |
| 2. | Вибір та обґрунтування схемотехніч-ного рішення | 03.03.2020 р. | |
| 3. | Розробка структурної схеми системи | 15.03.2020 р. | |
| 4. | Розробка функційної схеми системи | 20.04.2020 р. | |
| 5. | Опис алгоритму роботи системи. | 25.04.2020 р. | |
| 6. | Розробка блок-схеми програмного за-безпечення системи | 15.05.2020 р. | |
| 7. | Розробка функційної схеми системи | 20.06.2020 р. | |
| 8. | Аналіз балістичного відхилення робо-чого елемента. | 01.06.2020 р. | |
| 9. | Попередній захист дипломного проекту | 07.06.2020 р. | |
| 10. | Рецензування дипломного проекту | до 10.06.2020 р. | |
| 11. | Захист дипломного проекту | до 20.06.2020 р. | |

Студент _____ Віталій ШУЛІКОВ
(підпис) (ініціали, прізвище)

Керівник проекту _____ Юрій САМАРЦЕВ
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломного проєкту (роботи)

# АНОТАЦІЯ

У даному дипломному проєкті розробляється система тепловізійного прицілу. Система включає в собі високотехнологічні засоби, такі як сигнальний процесор ADV8003 та мікроконтролер BF537.

Протягом всього дипломного проєкту було спроектовано структурну, функціональну та принципову схеми пристрою, проведено аналіз та розрахунок похибок та розроблено програмне забезпечення.

Результатом дипломного проекту є система тепловізійного прицілу, яка значно розширює функціонал тепловізійного пристрою та виконує корегування параметрів за допомогою обробки параметрів навколишнього середовища. Система може бути використана у сферах, які пов’язані з мисливством чи у інших сферах, які потребують визначення точної траєкторії до вимірювального об’єкта у нічний період.

# ANNOTATION

This diploma project develops a system of a thermal imaging sight. The system includes high-tech tools such as the ADV8003 signal processor and the BF524 microprocessor.

Throughout the diploma project, the structural, functional and schematic diagrams of the device were designed, analysis and calculation of errors were performed and software was developed.

The result of the diploma project is the system of a thermal imaging sight, which significantly expands the functionality of the thermal imaging device and performs parameter adjustments by processing the parameters of the environment. The system can be used in areas related to hunting or in other areas that require the determination of the exact trajectory to the measuring object at night.

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»**

Кафедра інформаційно-вимірювальних технологій

ЗАТВЕРДЖУЮ

Зав. кафедрою ІВТ

_____ проф. Володимир ЄРЕМЕНКО

" ___ " _____ 2020 р.

# ТЕХНІЧНЕ ЗАВДАННЯ

**на дипломний проєкт**

«Система тепловізійного прицілу»

**ВМ61.300004.001 ТЗ**

УЗГОДЖЕНО:

Керівник дипломного проєкту

__Доцент_____

(Посада)

_____Самарцев Ю. М._____

(Прізвище І.ПБ.)

" ____ " _____ 2020 р.

Дипломник:

Ст. гр. ВМ-61-2

____Шуліков В.Ю.

(Прізвище І.ПБ.)

" ____ " _____ 2020р.

Залікова книжка __ТМ-6130

Київ 2020

# 1 ПРИЗНАЧЕННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Система тепловізійного прицілу – система, яка спрямована на розширення функціональних можливостей тепловізійного пристрою у нічний період доби. Система може бути використана у сферах, які пов’язані з мисливством чи у інших сферах, які потребують визначення точної траєкторії до вимірювального об’єкта.

# 2 ПІДСТАВИ ДЛЯ ВИКОНАННЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломний проєкт, видане і затверджене кафедрою інформаційно-вимірювальних технологій Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

# 3 МЕТА ТА ТЕХНІКО-Економічне ОБҐРУНТУВАННЯ РоЗРОБКИ

Метою розробки є створення телеметричної системи відео моніторингу складних природніх об’єктів, у вигляді тепловізійного пристрою з параметрами, відображеними у завданні на дипломний проект. Існуючі засоби для розпізнавання та моніторингу об’єктів мають недоліки у вигляді того, що або погано орієнтуються у темний період доби або не враховують залежність точності вимірювання, об’єкта моніторингу, від таких чинників, як навколишнє середовище та інше.

Впровадження обробки цифрових та аналогових сигналів в систему відео моніторингу об’єктів за допомогою сучасних засобів вимірювання та обробки даних, створює нові можливості для розширення функціоналу засобів моніторингу

# 4 ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки є:

- технічне завдання на дипломний проект;
- ГОСТи;
- науково-технічна література.

# 5 ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги, що визначають показники якості, технічні, метрологічні та експлуатаційні характеристики.

Телеметрична система повинна шляхом вимірювання необхідних даних, таких як відстань до об’єкту та температура навколишнього середовища, виконувати корегування зображення координат прицілу, та накладання цього зображення на сформований тепловізійний відеопотік, результат буде відображатися на відеодисплеї.

5.1.1 Система повинна мати наступні технічні характеристики.

5.1.1.1 Діапазон вимірювання далекоміром відстані до об’єкту: не більше ніж 1500м

5.1.1.2 Діапазон вимірювання температури навколишнього довища: -55 $^{o}C$ – +125 °C.

5.1.1.3 Кількість пікселей зображення: 800×600.

5.1.1.4 Система має використовувати композитний відеосигнал CVBS та стандарт відеопотоку BT.656.

5.1.1.5 Повинні бути присутні інтерфейси, як RS232, HDMI, USB для коректної взаємодії компонентів системи.

5.1.1.6 Діапазон робочих температур системи : -10$^{o}C$ – +40 $^{o}C$.

5.1.2 Вимоги до виду кліматичного виконання і вимоги до розроблюваної системи.

5.1.2.1 Система повинна бути тепло- та холодостійка.

Повинна бути не вразлива до механічних пошкоджень. Якщо ж була пошкоджена, то мала змогу і далі виконувати свої функції. Повинна зберігати свій зовнішній вигляд в процесі кліматичних впливів.

5.1.2.2 По стійкості до кліматичних і механічних впливів система повинна відноситись до групи 6 ГОСТ 22261- 94.

5.1.2.3 Система повинна зберігати працездатність, при характеристиках, наведених у таблиці 1.

Таблиця 1 - Характеристики

| Зовнішній фактор, що впливає | Нормальні умови функціонування |
|---|---|
| Вологість повітря | 95% при 95 °С |
| Атмосферний тиск, кПа (мм рт.ст.) | 60 – 106,7(480 - 800) |

## 6 ЕТАПИ РОЗРОБКИ

Етапи розробки, оформлення та узгодження дипломного проєкту наведено в табл. 6.1.

| № п/п | Найменування етапу | Дата |
|---|---|---|
| 1 | Розробка та узгодження технічного завдання | 12.05.2020 |
| 2 | Вибір та обґрунтування схемотехнічного рішення | 13.05.2020 |
| 3 | Розробка структурної схеми системи | 15.05.2020 |
| 4 | Розробка функційної схеми системи | 20.05.2020 |
| 5 | Опис алгоритму роботи системи. | 22.05.2020 |
| 6 | Розробка блок-схеми програмного забезпечення системи | 23.05.2020 |
| 7 | Розробка принципової схеми системи | 25.05.2020 |
| 8 | Аналіз балістичного відхилення робочого елемента. | 28.05.2020 |
| 9 | Розроблення програмного забезпечення дипломного проєкту | 01.06.2020 |

| | | |
|---|---|---|
| 10 | Оформлення графічних матеріалів | 02.06.2020 |
| 11 | Оформлення пояснювальної записки | 04.06.2020 |
| 12 | Попередній захист дипломного проєкту | 09.06.2020 |
| 13 | Рецензування дипломного проєкту | до 12.06.2020 |
| 14 | Захист дипломного проєкту | до 20.06.2020 |

Таблиця 6.1 – Етапи розробки дипломного проекту

Всі стандарти що використовуються в даному ТЗ на ДП є чинними на території України.

| №рядка | Формат | Познака | Найменування | Аркушів | № екз. | Примітки |
|---|---|---|---|---|---|---|
| 1 | | | Альбом 1 | | | |
| 2 | | | | | | |
| 3 | | | Документація загальна | | | |
| 4 | | | Заново розроблена | | | |
| 5 | А4 | ВМ61.300004.001 ТП | Відомість технічного проєкту | 1 | 1 | |
| 6 | А4 | ВМ61.300004.002 ПЗ | Пояснювальна записка | 65 | 1 | |
| 7 | А4 | ВМ61.300004.001 ТЗ | Технічне завдання | 5 | 1 | |
| 8 | | | | | | |
| 9 | А4 | ВМ61.300004.003 ПЕЗ | Система тепловізійного пристрою | | | |
| 10 | | | Перелік елементів | 2 | 1 | |
| 11 | | | | | | |
| 12 | | | Альбом 2 | | | |
| 13 | | | | | | |
| 14 | | | Графічна документація | | | |
| 15 | | | Розроблена заново | | | |
| 16 | А1 | ВМ61.300004.001 Е1 | Система тепловізійного пристрою | | | |
| 17 | | | Схема електрична структурна | 1 | 1 | |
| 18 | | | | | | |
| 19 | А1 | ВМ61.300004.002 Е2 | Система тепловізійного пристрою | | | |
| 20 | | | Схема електрична функційна | 1 | 1 | |
| 21 | | | | | | |
| 22 | А1 | ВМ61.300004.002 Е3 | Система тепловізійного пристрою | | | |
| 23 | | | Схема електрична принципова | 1 | 1 | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | ВМ61.300004.001 ТП |
| Зм. | Арк. | № докум | Під пис | Дата | |
| Розроб. | | Шуліков В.Ю. | | | Система тепловізійного прицілу<br>Відомість технічного проєкту |
| Перев. | | Самарцев Ю.М. | | | |
| Тех.контр. | | | | | |
| Н.контр. | | Богомазов С.А. | | | |
| Затвердж. | | Синиця В.І. | | | |

| Літ. | Арк. | Арк. |
|---|---|---|
| Т | 1 | 1 |

КПІ ім. Ігоря Сікорського
Каф. ІВТ, гр. ВМ61-2

# Пояснювальна записка
## до дипломного проєкту

на тему:  «Система тепловізійного прицілу»

Київ – 2020 року

Зміст

## Вступ

На сьогоднішній день існує достатня кількість засобів для розпізнавання та моніторингу об’єктів складної форми, що використовується для систем прицілу. Наприклад відеокамери спостереження та інше. Це дуже технологічні пристрої, але головний недолік цих засобів полягає у тому, що вони розкривають свій повний функціонал лише у світлий період доби. Іншими словами, у темний період доби у засобів виникають труднощі при розпізнаванні та моніторингу об’єктів складної форми. Тому використовують такі засоби, як тепловізори, для розширення функціональних можливостей моніторингу та розпізнавання об’єктів у темний період доби. Але, при використанні тепловізорів в якості прицілів, вони знову ж таки мають ряд недоліків. Керуючись подібними прицілами, траєкторія руху кулі при пострілі буде спотворена. Причина такого спотворення полягає у тому, що тепловізійні приціли не враховують залежність точності розпізнавання параметрів від таких параметрів, як дальність до об’єкта, температури навколишнього середовища, швидкості вітра та інших погодних умов. Тобто на кулю у польоті впливає багато чинників, наприклад сила гравітація, тому у результаті ми отримаємо досить неточну траєкторію польоту кулі.  Для того ,щоб можна було врахувати ці параметри, які вносять корективи на траєкторію польоту кулі, було вирішено розробити систему, яка б дозволяла вводити коригуючі параметри на зображення відео об’єкту, які дозволяють використовувати для прицілу не первинне зображення на екрані, а той результат вимірювань, який буде отриманий за допомогою наших вимірювальних засобів. Тобто, на первинне зображення, яке буде формувати тепловізійний пристрій буде накладена прицільна сітка, з попередньо скорегованими координатам прицілу та статичне зображення, у вигляді тексту. У результаті система підвищує точність вимірювання параметрів у нічний період доби та максимально спрощує всі складності при формуванні точної траєкторії польоту кулі.

# 1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

## 1.1 Тепловізійний пристрій

Для системи тепловізійного прицілу, тепловізійний пристрій буде використовуватися і як формувач растрового зображення так і засіб, між яким буде вбудовано інші компоненти. Тому габарити тепловізійного пристрою дуже важливі для нашої системи. Перейдемо до розглядання тепловізора, як окремого засобу.

Схожий на відеокамеру, тепловізор реєструє тепло[1]. Усі предмети випромінюють тепло в інфрачервоному спектрі, який є невидимим для ока людини. Чим більша температура, тим сильніше це випромінювання. Прилад створює теплову карту - контрастне зображення на екрані, на основі результату різниці нагріву ділянок фону  і поверхні. Живі істоти і деталі рельєфу, навіть в повній темряві, стають добре помітні.

Болометричний сенсор, який чутливий до теплового випромінювання - основа тепловізійного прицілу. Мікроболометр є напівпровідниковою матрицею з терморезисторів, які у свою чергу напиленні на кристал. Відстань між елементами не більше 20 мікрон, а чутливість - соті долі градуса. США, Франції і Китай, на даний момент, основні виробники[1].

Теплове випромінювання не пропускається звичайним склом, тому об'єктиви прицілів зроблені з рідкоземельного монокристалічного германію. Інфрачервона оптика має чорний колір і характерний блиск - для захисту від ушкоджень германієві лінзи покривають тонким і міцним шаром синтетичного алмазоподобного матеріалу, так званого DLC- вуглецю.

Власне матриця (її матеріал, розмір і межа дозволу) і розмір германієвого об'єктиву визначають вартість тепловізійних прицілів. Окуляри тепловізійних прицілів роблять з оптичного скла, з діоптрійним налаштуванням і м'яким наочником. Окуляр фокусується на екран - світлодіодну матрицю високого розділення. У більше ранніх, дешевих моделях застосовувалися ЖК-екрани, але

для більшості мисливців вони не підходять, тому що рідкі кристали бояться низьких температур. Технології виготовлення екранів сильно випереджають технології мікроболометрів. Тому, попри те, що екран впливає на вартість прицілу, але не так істотно як сенсорна матриця і розмір об'єктива[1].

Так влаштовано більшість тепловізійних приладів. На сьогоднішній день існує декілька варіантів мисливських тепловізорів.

- Біноколь
- Монокуляр. Можна використовувати без установки на рушницю, саме тому це найрозповсюдженіший тип тепловізорів для полювання.
- Насадка
- Приціл. Має кріплення для посадки на рушницю та прицільну сітку. Тому саме приціл призначений для ведення вогню. Він жорстко кріпиться на зброю і не боїться навантажень при віддачі.

Сенсор - найважливіший елемент. Є мікроболометричною матрицею з щонайменших часток - абсорберів. Абсорбери мають властивість поглинати електромагнітні хвилі випромінювання. Чим більше поглинання, тим більше опору і нагрів болометра, що у свою чергу переводиться в цифровий сигнал[1].

Розширення матриці. Мікроболометричні сенсори розрізняються кількістю чутливих елементів. Зазвичай застосовують основні розширення матриці стандарту VGA 640x480 і QVGA 384x288. Чим вище дозвіл, тим чіткіше зображення на екрані і точніше сам приціл. Є навіть сенсори 1024x640, але вони дорогі для цивільного користування. Кратність об'єктиву, оптичне збільшення. Число перед знаком "x" показує, в скільки разів зображення в прицілі більше, ніж видиме простим оком. Фіксованого збільшення в 1-4 рази (об'єктивів 1x - 4x) досить для полювання на великих тваринах[1]. Тут йдеться саме про оптичне збільшення, тому що багато моделей мають цифровий зум, який по суті програмно розтягує пікселі. Поле зору. Кут огляду, який можна спостерігати без переміщення точки прицілювання. Обернено пропорційний до кратності об’єктиву. Чим вище кратність, тим нижче поле зору і контрастність, тому при великій кратності

необхідно застосовувати об'єктиви більшого діаметру, а це сильно впливає на вартість прицілу.

Частота кадрів. Оновлення зображення в окулярі зору відбувається циклічно. Чим вище частота кадрів, тим коротший час затримки. Це особливо важливо при стрільбі по рухомих цілях. Таким чином, частота кадрів екрану, а також реакція пікселів не повинна бути меншою або перевищувати можливості мікроболометричного датчика[1].

Якщо порівнювати технічні характеристики, вони приблизно однакові для більшості сучасних пам’яток. Справа в тому, що в основі інструментів лежать ті самі мікроболометри французької компанії ULIS.

Як вибрати, чи оптика, мікроболометри та частота кадрів однакові? Виробники вказують на діапазон - три значення, які відрізняються від моделі до моделі:

Усі ці величини - обчислюються, з вірогідністю не більше 50%.Для мисливця має значення саме дистанція ідентифікації. З цієї відстані можна говорити про прицільну стрільбу.

Дальність точного пострілу значно менше дистанції ідентифікації. Вона залежить від погоди, температури, типу мети, ландшафту, балістики зброї, але в першу чергу, від уміння і досвіду самого мисливця[1].

У виробників тепловізійних прицілів різний підхід до трактування дистанції ідентифікації мети і пострілу. Втім, і атмосферні умови додають невизначеності. Проте, з упевненістю можна сказати, що дистанції виявлення біологічних цілей і точного пострілу навіть у недорогих тепловізійних прицілів не нижчі, ніж у найсучасніших класичних нічних прицілів[1].

Конкуренція на ринку тепловізійних прицілів висока. У рік з'являється по дві-три моделі з новими можливостями і поліпшеними характеристиками. Частина доповнень - просто реклама, але деякі речі дійсно корисні на полюванні: Заповнений азотом корпус. Перепади температури можуть привести до утворення конденсату і виходу з ладу мікроелектроніки. Щоб уникнути цього, корпуси прицілів, прийнято робити герметичними і заповнювати азотом. Раніше це були тільки армійські зразки, але практично усі сучасні моделі створюють

герметичними. Безшумна калібровка. Болометр нагрівається в процесі роботи і виникають перешкоди. Їх усувають калібруванням. Для цього можна закривати матрицю шторками, як у фотоапараті. При цьому виникає характерне клацання, схоже на спуск затвора. Цей звук видає снайпера і мисливця. Порівняно недавно з'явилася тиха технологія калібрування, без використання шторок, на яку перейшли усі виробники тепловізійних прицілів[1].

Закон забороняє використати армійські приціли на мисливській зброї. Це і не треба, тому що багато моделей на ринку - вироби подвійного призначення. Вони створені за замовленням збройних сил і силовиків. Наприклад, тепловізійний приціл ПТ- 3 - чисто армійська розробка, доступна в цивільному виконанні.

Комерційні моделі ніколи не стояли на озброєнні і сконструйовані спеціально для мисливців. Вони нічим не гірші за армійських, а за деякими параметрами навіть перевершують. Це стосується запису відео, програмних опцій і зручності пристрілки. По рівню надійності і непримхливості можуть поступатися військовим моделям.

У бюджетних тепловізійних прицілах виробники економлять на всьому: розмірі і дозволі матриці сенсора, розмірі об'єктиву з дорогого германію, екрані для проекції зображення, морозостійкості і герметичності приладу[1].

Для прикладу приведемо  пару тепловізійних пристроїв, які досить популярні на сьогоднішній день.З бюджетної лінійки тепловізійних монокулярів можем виділити пристрої фірми Helion від компанії Pulsar. Бюджетна ціна та зручний інтерфейс це головні плюси цієї лінійки.

Helion XQ28(Рис 2.1) яскравий представник тепловізійних монокулярів.

Рисунок 1.1.2 - Helion XQ28

Особливості монокуляра

Для знімання даних важливо, щоб мікроболометр не враховував власну теплоту приладу[2]. Для вирішення цієї проблеми на корпусі виготовлені прорізи для відведення тепла.Висока частота зміни кадрів. Після отримання спектральних даних важливо враховувати з якою частотою вона з'являється на дисплеї. Helion XQ28 має частоту 50 ГЦ, що повністю задовольняє вимоги при перегляді об'єкту, що рухається. Головні переваги:

- Змінне збільшення. Прилад має плавне збільшення.
- Відеозапис і Wi - Fi - модуль. В якості нововведення в Helion був доданий відеорекодер, який записує все, що відбувається перед очима спостерігача у внутрішню пам'ять. Після чого, користувач може перенести усі дані на пристрої через USB- порт. Дані можна перенести і через Wi - Fi. Наявність модуля дозволяє вести трансляцію на екрані пристрою і міняти налаштування через додаток.
- Вологостійкість і міцний корпус. Для багатьох завдань при використанні тепловізора важливим чинником буде міцність корпусу. Helix XQ28 виготовлений з щільного пластика і має клас захисту IPX7 (допускає короткочасне занурення у воду).

– Дистанційний пульт управління. Зручність використання ПУ оцінять мисливці, які можуть годинами вичікувати виходу звіра, при цьому завжди мають бути напоготові.
– Стадіометричний далекомір. У своїй складній назві функція представлена у вигляді двох шкал. На вибір дається три об'єкти з різною висотою. Перша шкала виставляється по нижній частині об'єкту спостереження, друга - по верхній. Залежно від положення шкал на екрані відображаються ці відстані до об’єкта[2].

Наступний тепловізійний приціл моделі Dedal Venator(Рис. 1.1.2). Тепловізійний приціл Venator є цілим комплексом для рішення завдань ведення стрільби на різну відстань. Велику прицільну дальність забезпечує об'єктив F50/1.2 і унікальне програмне забезпечення прицілу. У приціл вбудованно балістичний калькулятор, який враховує поправки при веденні стрільби[3].

Рисунок 1.1.2

– Основні властивості приладу Venator.
– Легке і компактне виконання.
– Дальність виявлення цілі складає 1800 метрів.
– Низьке енергоспоживання (до 8 годин роботи).
– Ударна стійкість 6000 Дж.
– Автоматичне електронне калібрування.
– Різні режими зображення.
– Наявність балістичних прицільних сіток.
– Можливість підключення відеорекодера і зовнішнього живлення[3].

Висновок: Було розглянуто приклади тепловізійних прицілів та їх характеристик. З отриманою інформацією можна зробити вибір.

### 1.2 Далекомір

Наступний крок це визначення далекоміра, який буде підходити до нашого проекту. Але спочатку дізнаємося більше про види далекомірів.

Взагалі далекомір - це прилад, призначений для визначення відстані між спостерігачем і видаленим об'єктом без необхідності наближатися до нього[4]. Він широко використовується в геодезії, а також у будівництві, топографії і інших сферах. Також далекомірами користуються військові для коригування вогню з снайперської зброї і мінометних установок. За принципом роботи існуючі конструкції далекомірів розділяють на активні та пасивні. Що стосується пасивних, то вони не посилають ніяких сигналів. Визначення відстані здійснюється за абсолютно іншим принципом. Такі інструменти працюють за законами геометрії. За допомогою пасивних приладів здійснюється обчислення побудованого рівнобедреного трикутника, за параметрами якого можна вирахувати відстань.

Нас цікавлять саме активні, які наводяться об'єктивом на точку, до якої необхідно виміряти відстань, після чого відправляють на неї світловий або звуковий сигнал. Досягнувши поверхні предмета, той відбивається і повертається назад. Чутливий елемент приладу уловлює хвилю і розраховує відстань до об'єкту на основі часу, який пішов на її пересування[4].

Активні далекоміри бувають звукові, сітлові та лазерні

Ультразвуковий далекомір (Рис 1.2.1) є самим неточним пристроєм, працюючим за активним принципом. Це устаткування має схожий метод з тим, що використовують для орієнтування дельфіни або кажани.


Рисунок 1.2.1- модуль ультразвукового далекоміра

Прилад створює звукову хвилю, спрямовану вперед на об'єкт, до якого треба поміряти відстань. При досягненні імпульсом перешкоди створюється луна, яка відбивається і потрапляє на чутливу частину ультразвукового пристрою.

Такі прилади використовують звук з високою частотою близько 40 Кгц. Він невловимий для вуха людини, тому застосування подібного далекоміра не викликає ніякого дискомфорту. Це порівняно недорогі пристрої, але щоб ними скористатися, необхідно правильно направити імпульс, на що йде час[4].

Лазерний далекомір (Рис 1.2.2) використовує лазерний промінь для визначення відстані до об'єкта[5]. Найпоширеніша форма лазерного далекоміра працює за принципом часу польоту , посилаючи лазерний імпульс у вузькому промені до об'єкта і вимірюючи час , який імпульс відбиває від цілі, і повертається відправнику. Через високу швидкість світла ця методика не підходить для вимірювань високої точності субміліметра, де часто застосовують триангуляцію та інші методи.


Рисунок 1.2.2 – приклад лазерного далекоміра

Незважаючи на те, що промінь вузький, він з часом поширюється на великі відстані через розбіжність лазерного променя, а також за рахунок сцинтиляції та ефекту блукання пучка, спричиненого наявністю в повітрі бульбашок , що діють як лінзи[5].

Ці атмосферні спотворення в поєднанні з розбіжністю самого лазера і з поперечними вітрами, які служать для виштовхування атмосферних бульбашок тепла в бік, можуть поєднуватися, щоб ускладнити точне зчитування відстані предмета, скажімо, під деякими деревами або за кущами, або навіть на великі відстані більше 1 км у відкритій і неприкритій пустельній місцевості.

Частина лазерного світла може відбиватися від листя або гілок, які знаходяться ближче до об'єкта, що дає раннє повернення і занадто низьке читання. Крім того, на відстані 365 м більше, ніж на відстані 1200 футів, ціль, якщо знаходиться в безпосередній близькості до землі, може просто зникнути в міраж , викликаний градієнтами температури повітря в близькості до нагрітої поверхні, що вигинає лазерне світло[5]. Усі ці наслідки потрібно враховувати.

Відстань між точкою A і B задається числом (Рис 1.2.3)

$$D = \frac{ct}{2}$$

де $c$ - швидкість світла, а $t$ - кількість часу на зворотну подорож між A і B.

$$t = \frac{\phi}{\omega}$$

де $\varphi$ - фазова затримка, що досягається світлом, що рухається, а $\omega$ - кутова частота оптичної хвилі[5].

Потім підставляючи значення в рівняння,

$$D = \frac{1}{2}ct = \frac{1}{2}\frac{c\phi}{\omega} = \frac{c}{4\pi f}(N\pi + \Delta\phi) = \frac{\lambda}{4}(N + \Delta N)$$

У цьому рівнянні $\lambda$ - довжина хвилі $c$ / $f$; $\Delta\varphi$ - частина фазової затримки, яка не відповідає $\pi$ (тобто $\varphi$ модуль $\pi$ ); $N$ - ціле число півциклів хвилі обертання, а $\Delta N$ - решта дробової частини.

Час польоту - це вимірювання часу, необхідного для імпульсу, який проходить до цілі та назад. З відомою швидкістю світла та точним вимірюванням часу, який можна провести, відстань можна обчислити. Багато імпульсів спрацьовують послідовно і найчастіше використовується середня реакція. Ця методика вимагає дуже точної схеми синхронізації наносекунди.

Фазовий зсув декількох частот - це вимірювання зсуву фази декількох частот на відбиття, потім вирішує деякі одночасні рівняння, щоб дати остаточну міру[5].

Лазерні далекоміри є електронним пристроєм, який потребує джерела живлення. Ним може виступати вбудована акумуляторна батарея або звичайні пальчикові батареї. У плані економії кращий пристрій на акумуляторі, який можна заряджати від електромережі[4]. Собівартість забезпечення його роботи набагато нижча, ніж при періодичній купівлі батарей для зміни. Важливою перевагою, яку має лазерний далекомір, є можливість виміру відстані до певної точки. Інструменти інших типів такої функції не мають. Пучок лазерного променя дуже тонкий, тому він доходить до необхідної ділянки об'єкту і відбивається від нього назад. Якщо поверхня є рельєфною, приміром прямовисна скеля, то тільки такий пристрій дасть можливість отримати точні данні.

Також існує стадіометричний далекомір, який в основному вже є вбудованим у всі сучасні тепловізори. Іншими словами це програмний далекомір, який має алгоритм розрахунку дістанції за допомогою вже відомих даних росту та типом істоти та прицільної сітки зі шкалою. Принцип роботи далекоміра наведений на рисунку 1.2.3. Ключовим фактором являється правильний підбір росту тварини, для більшої точності знаходження дістанції. Але в темний період доби буде проблематично визначити тип та ріст тварини. Тому не завжди цей далекомір є актуальний[6].

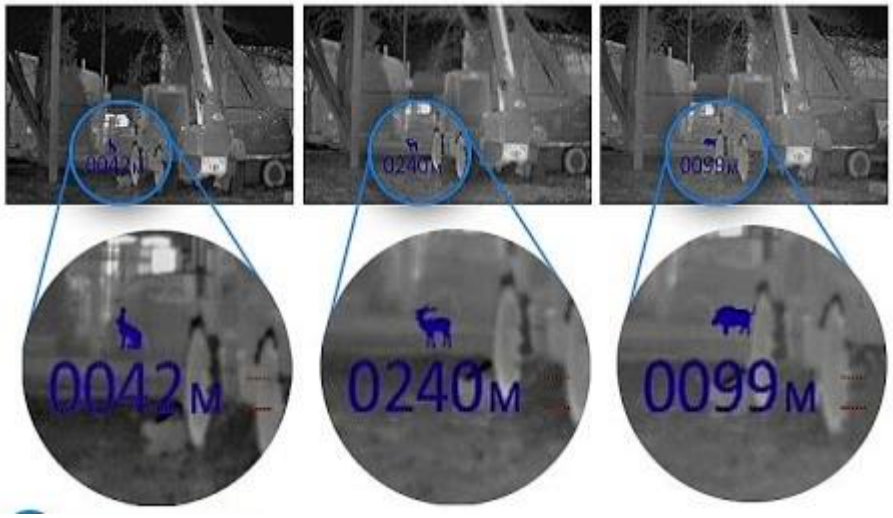


Рисунок 1.2.3 – принцип роботи стадіометричного далекоміра

Було принято рішення використовувати лазерний далекомір, якщо більш точно то саме модуль лазерного далекоміра. Пропоную розглянути існуючі варіанти. Лазерний далекомір з датчиком відстані 100 м/328фт, модуль(Рис 1.2.4) Diastimeter, пропонує поодинокий та безперервний вимір. Основна характеристики[7].

- Nterface: 2,5-2,8 v TTL сигнал від PC.
- Трипровідний послідовний інтерфейс (мікрокомп'ютерний UART інтерфейс з одним чіпом), 19,2 кГц
- 2 мА електричний потік: DC2.5-2.8v, режим очікування, вимір 120 мА
- Тип лазера : 635 нм, 1 мВт (червоний)
- Діапазон: від 0,02 до 100 метрів

- Швидкість виміру : 0,3-3 s-Стандартна точність: - 2 мм
- Температура зберігання : - 20-60 температура використання : - 10-40
- Розмір модуля : прибл.72*40*18 Мммм (без екрану)
- Вага виробу : прибл.: 70 г

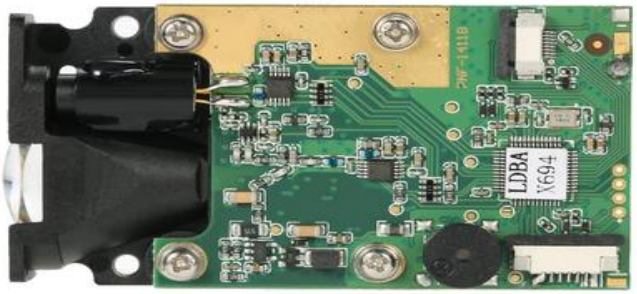


Рисунок 1.2.4 – Модуль далекоміра Diastimeter

Наступний приклад модель Serials Output LRF Module(Рис 1.2.5)

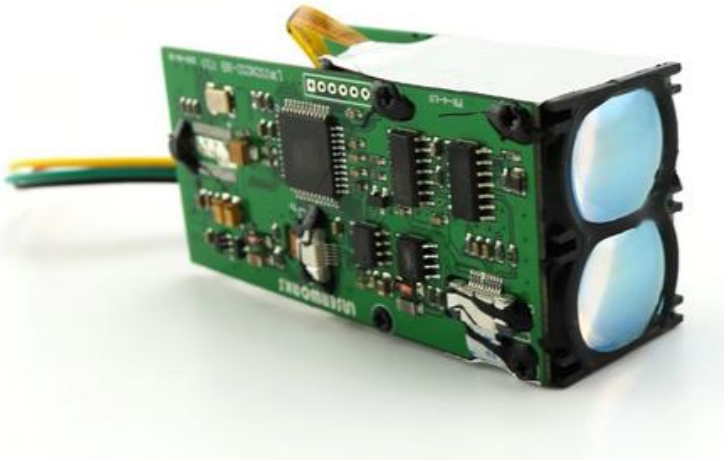

Рисунок 1.2.5 – Модуль далекоміра Serials Output LRF

- Діапазон потужності: від 5 до 600 метрів.
- Розмір: 59*30*34 мм.
- Вага: 37,5 г.
- Точність: ± 0,5 метра.
- Інтерфейс передачі данних: RS232 або TTL.
- Робоча напруга 3-5 В.

– Робоча температура: от- 20 до 60 °C.
– Лазер: клас 1, безпечний для очей людини.
– Довжина хвилі: 90 нм.
– Частота: 1 гц[8].

### 1.3 Датчик температури

Дальність траєкторії кулі залежить не лише від сили земного тяжіння, але й від навколишнього середовища. А саме від густини повітря, яка може змінюватися за рахунок зміни температури, вологості або атмосферного тиску[9]. За табличні дані було взято:

– атмосферний тиск 750мм
– температура повітря +15°C
– вологість повітря 50%
– повна відсутність вітру

Відходження від табличних даних буде впливати на політ та траєкторії кулі. Густина повітря буквально описує, наскільки щільне або густе повітря. Висока температура і низький тиск знижують щільність повітря, низькі температури та підвищений тиск збільшують щільність. Тому при підвищені температури повітря при спеці помітно підвищується траєкторія, і навпаки, в мороз щільність повітря помітно підвищується і кулі йдуть набагато нижче[9].

Також потрібно пам’ятати що куля сповільнюватиметься швидше у густому повітрі. Чому це має значення? Куля, яка сповільнюється, досягне цілі за більший проміжок часу , тому буде більш схильна до впливу вітру та земного тяжіння. Саме тому потрібно підключити датчик температури до нашого пристрою. Розглянемо основні види датчиків.На сьогодні випускаються наступні види температурних датчиків :

Термістори - це термометри опору, виготовлені на основі змішаних оксидів перехідних металів. Термістори діляться на два основні типи - PTC (з позитивним коефіцієнтом) і NTC (з негативним коефіцієнтом температурного опору). Найбільш поширені температурні датчики NTC[10]. Термістори PTC(Рис 1.3.1)

застосовуються виключно у вузьких діапазонах температур (всього декілька градусів), тому, їх використання частіше обмежується системами контролю і сигналізації. В цілому, термометри дуже чутливі до вимірюваної температури, але, на жаль, цього не можна сказати про лінійність вихідного сигналу.


Рисунок 1.3.1 – Зовнішній вид термістора

Термопари. Це устаткування є ідеальним рішенням для виміру температури в максимально широкому діапазоні (до 2300°C). Воно відрізняється високою точністю і вопроизводимостью. Але, важливо відмітити: термопари потребують установки схем посилення, що потрібне для його подальшої обробки[10].

Терморезистерні датчики**(**Рисунок 1.3.2). Принцип роботи терморезистивных датчиків температури (RTDs - Resistance Temperature Devices) грунтований на пропусканні через них електричного струму.

Рисунок 1.3.2 – Зовнішній вид терморезисторного датчика

Напівпровідникові датчики. Сучасні напівпровідникові датчики виконують свої функції в широкому діапазоні температур. Вони мають високу точність.

Пристрої оснащені вбудованою схемою посилення сигналу, що дозволяє настроювати устаткування на необхідну температурну залежність[10]. Розрізняють датчики температури і за матеріалом виконання чутливого елементу. Існують:

- датчики з платиновим чутливим елементом;
- корпусовані датчики;
- датчики з напівпровідниковим чутливим елементом.

Огляд цифрового датчика температури DS18B20(Рис 1.3.3). Це повноцінний цифровий термометр, здатний вимірювати температуру в діапазоні від -55С до 125С з програмованою точністю 9-12 біт[11].

.

Рисунок 1.3.3- Датчик температури DS18B20

При виготовленні на виробництві, кожному датчику привласнюється свій унікальною 64-бітова адреса, а обмін інформацією з провідним пристроєм (мікроконтроллером або платою Arduino) здійснюється по шині 1 - wire. Такий підхід дозволяє підключати до однієї лінії цілу групу датчиків, аж до 264.

Для того, щоб зрозуміти принципи спілкування з датчиком, необхідно ознайомитися з його внутрішньою складовою. Ця на вигляд маленька мікросхема містить в собі цілий ряд електронних блоків і модулів. На Рисунку 1.3.4 показана структурна схема датчика DS18B20.

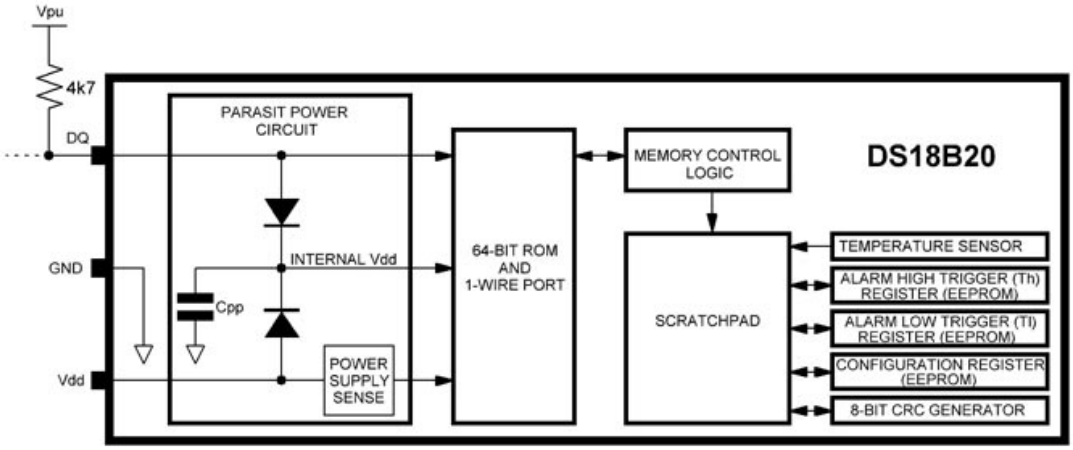


Рисунок 1.3.5 – Структурна схема датчика DS18B20

Перше на що хотілося б звернути увагу - це спосіб живлення мікросхеми. Їх всього два: режим прямого харчування (напруга подається на виведення Vdd і GND) і режим паразитного харчування (датчик живиться від лінії даних). Детальніше ці режими будуть розглянуті нижче в статті.

Із структурної схеми видно, що за пряме живлення відповідає блок (POWER SUPPLY SENSE), а режим паразитного харчування забезпечує блок (PARASIT POWER CIRCUIT)

Далі слідує модуль "64 - BIT ROM AND 1 - WIRE PORT". У нім міститься інформація про унікальну адресу кожного пристрою. Як було сказано вище цей код записується заводом-виготівником у внутрішню енергонезалежну пам'ять датчика (EEPROM) і ніколи не повторюється. Тут же розміщена і підсистема взаємодії з 1 – Wire інтерфейсом[11].

Блок "MEMORY CONTROL LOGIC" здійснює взаємозв'язок між командами інтерфейсу 1 - Wire і внутрішньою пам'яттю датчика SCRATCHPAD. Ця пам'ять, у свою чергу, взаємодіє з декількома спеціалізованими регістрами DS18B20, а саме:

- TEMPERATURE SENSOR. Дозволяє зчитувати перетворене значення температури.

- ALARM HIGH TRIGGER і ALARM LOW TRIGGER. Ці регістри дозволяють виставити верхній і нижній поріг спрацьовування сигналів тривоги при виході температури за вказані межі.

-CONFIGURATON REGISTER. Цей регістр призначений для налаштування роздільній здатності температурного датчика. Регістр може бути конфігурований на виконання перетворення температури з точністю 9 біт, 10 біт, 11 біт або 12 біт, що відповідає точності виміру 0.5oC, 0.25oC, 0.125oC і 0.0625oC.

- 8 - BIT CRC GENERATOR. Цей регістр призначений для генерації контрольної суми з метою підвищення захисту даних[11].

### 1.4 Балістична таблиця

Далі розглянемо балістичні таблиці, які будуть оброблятися мікроконтролером на основі даних відстані та температури. Взагалі балістика це наука про рух тіл, кинутих в просторі, грунтована на математиці і фізиці. Вона займається, головним чином, дослідженням руху снарядів, випущених з вогнепальної зброї, ракетних снарядів і балістичних ракет. Розрізняють внутрішню балістику, що займається дослідженням руху снаряда в каналі знаряддя, в протилежність зовнішній балістиці, що досліджує рух снаряда після виходу зі знаряддя. Під зовнішньою балістикою розуміють, як правило, науку про рух тіл в повітряному і безповітряному просторі під дією тільки зовнішніх сил[12].

В балістичній таблиці міститься набір даних щодо зовнішньої балістики боєприпасів або, якщо точніше, конкретної кулі, вистріленої в певних умовах. Дані, що містяться в таблиці, дозволяють легше обчислити коригування прицілу, необхідні для стрільби на відстані.

Таблиця містить як мінімум декілька типів даних (зміст колонок таблиці може варіюватися залежно від різновиду виду стрільби) - це дистанція в метрах або ярдах, вертикальна поправка і бічний знос вітром. Часто балістичні таблиці містять ще і інші дані, наприклад швидкість кулі на підльоті до відповідної дистанції, час польоту в секундах до відповідної дистанції.

Балістична таблиця складається конкретно для кожного патрона. А якщо бути точним, то для системи ствол/патрон[12]. Тому, що куля від одного і того ж патрона, але із стволів різної довжини і з різним кроком нарізу полетить природно по-різному.

Також таблиця складається у відповідності пристрілки зброї на певну дистанцію. При зміні дистанції пристрілки зброї в нуль, потрібно міняти і усю таблицю. Враховується і властивості холодного і гарячого стволів. Стрілкам, яким важливий перший постріл, має сенс створити таблиці під обидва стволи, тому, що вони дають різні точки влучення. Одиницею виміру вертикальних поправок у балістичній таблиці служить, як правило кутова хвилина[12].

| *Дистанция* | *0* | *50* | *100* | *150* | *200* | *250* | *300* | *350* | *400* | *450* | *500* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Пристрелка на 100 м | - | -0.3 | 0 | 0.7 | 1.8 | 3.1 | 4.6 | 6.2 | 8.1 | 10.2 | 12.6 |
| Пристрелка на 200м | - | -1.5 | -1.8 | -1.1 | 0 | 1.3 | 2.8 | 4.4 | 6.3 | 8.4 | 10.8 |
| Пристрелка на 300м | - | -4.3 | -4.6 | -3.8 | -2.8 | -1.5 | 0 | 1.7 | 3.5 | 5.6 | 8.0 |

Рисунок 1.4.1 -Приклад таблиці поправок в МОА

Для перекладу кутових хвилин в клацання прицілу, потрібно лише знати вартість клацання і просто помножити хвилини на цю вартість. Іноді у балістичних таблицях вказуються не поправки, а зниження траєкторії кулі на певних дистанціях, наприклад в сантиметрах.

| **Дистанция** | 0 | 50 | 100 | 150 | 200 | 250 | 300 | 350 | 400 | 450 | 500 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Пристрелка оружия на 100 м | +4.0 | +0.2 | 0 | -3.8 | -12.7 | -27.8 | -50.5 | -82.7 | -127.0 | -186.4 | -264.3 | **см** |

Рисунок 1.4.2 - Приклад таблиці зі зниженням в сантиметрах

Тому, якщо в таблиці вказані не хвилини, а сантиметри зниження, то тут для перекладу в хвилини потрібно виконати наступну дію:

M = P / (1m * D)

де M - шукана поправка в хвилинах;

P - зниження в сантиметрах;

1M - розмір однієї кутової хвилини на 100 метрів = 2.65 см;

D - кількість сотень метрів до мети.

Чи:

M = P / 1m / D

Що одно і теж.

Після цього, вже знайомою дією хвилини переводяться в кліки барабанчика. Одиницею ж виміру бічних поправок на вітер, у балістичній таблиці служать сантиметри або дюйми, а не хвилини.

Будь-яка балістична таблиця робиться під конкретні погодні умови. Адже температура, вологість, тиск і висота над рівнем моря теж мають значення. Це як правило погодні умови, в яких найчастіше працює стрілець. Але є таке поняття, як стандартні погодні умови. І якщо немає спеціальних погодних умов, то таблиця складається по стандартних: 59º F (температура); 78um (вологість); 29,53 inches (тиск 760 мм рт.стлб); 500 feet (висота)[12].

BALLISTIC TABLE IN METERS FOR
**7,62mm M80 Ball, 148 gr FMJBT for 24" Barrel**
100 yd. Zero, 59 deg F., Elevation ASL, Press 29,53 in, Hum 78%
BC's: .404 (H), .404 (M), .404 (L)

| Range METERS | Velocity (fps) | Maximum Ordinate (inches) | Bullet Path (inches) | Time of flight (sec) | Come Up to range (MOA) | Come Up cumulative (MOA) |
|---|---|---|---|---|---|---|
| 0 | 2854 | | | | | |
| 100 | 2606 | +.2 | Zero | 0.1 | 0 | 0.00 |
| 200 | 2372 | +2.4 | -4.6 | 0.3 | 2.5 | 2.5 |
| 300 | 2149 | +6.9 | -16.6 | 0.4 | 3 | 5.5 |
| 400 | 1939 | +19.3 | -37.7 | 0.6 | 4 | 9.5 |
| 500 | 1742 | +25.5 | -69.8 | 0.7 | 4.5 | 14 |
| 600 | 1562 | +41.6 | -115.6 | 0.9 | 5 | 20.5 |
| 700 | 1400 | +64.3 | -178.5 | 1.0 | 6.5 | 25.5 |
| 800 | 1260 | +95.6 | -262.7 | 1.5 | 7.5 | 33 |
| 900 | 1147 | +137.9 | -373.2 | 1.7 | 8.5 | 41.5 |
| 1000 | 1063 | +193.6 | -515.6 | 2.0 | 10 | 51.5 |

Рисунок 1.4.3 - Приклад таблиці для гвинтівки с 24 дюймовим стволом для патрону 7,62мм.

## 1.5 Мікроконтроллер

Мікроконтроллер це невелика мікросхема, яка об’єднує в собі процесор, пам'ять (ПЗП і ОЗУ), периферійні пристрої, змусили їх працювати і взаємодіяти між собою і зовнішнім світом за допомогою спеціальної мікропрограми, яка зберігатися усередині мікроконтроллера. Основне призначення мікроконтроллерів - це управління різними електронними пристроями. Популярністю у розробників користуються 8-бітові мікроконтроллери PIC фірми Microchip Technology і AVR фірми Atmel, 16-бітові MSP430 фірми TI, а також 32-бітові мікроконтроллери, архітектури ARM, яку розробляє фірма ARM Limited і продає ліцензії іншим фірмам для їх виробництвах[13].

Мікроконтролер характеризується великим числом параметрів, оскільки він одночасно є складним програмно-керованим пристроєм і електронним приладом (мікросхемою). Приставка "мікро" в назві мікроконтроллера означає, що виконується він за мікроелектронною технологією. В ході роботи мікрконтролер прочитує команди з пам'яті або порту введення і виконує їх. Що означає кожна команда, визначається системою команд мікроконтроллера. Система команд закладена в архітектурі мікроконтролера і виконання коду команди виражається у проведенні внутрішніми елементами мікросхеми визначених мікроопераций[13].

Візьмемо за приклад модель МК BF537(Рис 1.5.1) та розглянемо більш детально.

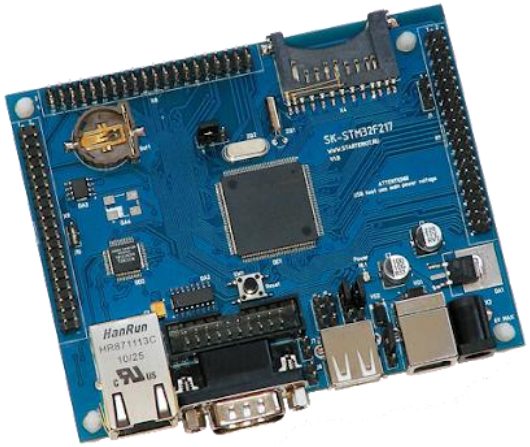


Рисунок 1.5.1 - модель МК BF537

Процесор ARM Cortex-M3 - це ARM-процесори останнього покоління для вбудованих систем. Він був розроблений для того щоб забезпечити не дуже дорогу платформу, що відповідає потребам впровадженими MCU, зі зменшеним числом контактів та низьким використанням енергії, забезпечуючи при цьому обчислювальні характеристики та розширену відповідь на перебої[14].

Завдяки вбудованому ядру ARM сім'я BF537 сумісна з усіма програмними забезпеченнями та інструментами ARM.Адаптивний прискорювач пам'яті в реальному часі (ART AcceleratorTM)ART AcceleratorTM - це інструмент для прискорення пам'яті, так званий прискорювач. Щоб випустити процесор на повну 150 DMIPS продуктивність на цій частоті, прискорювач реалізує чергу попереднього вибору інструкцій та кеш гілки, що збільшує швидкість виконання програми із 128-бітною флеш-пам'яттю. На основі критерію CoreMark, продуктивність, досягнута завдяки прискорювальнику ART, еквівалентна виконанню стану програми 0 очікування з флеш-пам'яті на частоті процесора до 120 МГц.

Пристрої BF537 вбудовують 128-бітну флеш-пам’ять 128 Кбайт, 256 Кбайт, 512 Кбайт, 768 Кбайт або 1 Мбайт, доступний для зберігання даних чи програм. На пристроях також є 512 байтна пам'ять OTP, які можна використовувати для зберігання важливих користувацьких даних, таких як MAC адреси Ethernet або криптографічні ключі. DMA-контролер (Контролер Прямого

доступу до пам'яті)Пристрої мають два DMA, тобто два прямих доступи до пам'яті з загальним призначенням (DMA1 та DMA2) з 8 потоками кожен[14]. Вони здатні керувати передачею з одної пам'яті до іншої, периферійною пам’яттю та переходом пам’яті до периферії. Вони поділяють деякі централізовані FIFO тобто алгоритм заміщення комірок пам'яті для периферійних приладів APB (перифирійна шина)/ AHB (вдосконалена високопродуктивна шина), підтримують передачу пакетних передач і призначені для забезпечення максимальної периферійної пропускної здатності для (AHB / APB).Два контролери DMA підтримують кругове управління буфером, так що конкретний код не потрібен, коли контролер доходить до кінця буфера. Два контролери DMA також мають функцію подвійної буферизації, яка створює автоматизування та об'єднання двох буферів пам'яті, не вимагаючи спеціального коду для цього. Кожен потік підключений до апаратних запитів DMA з підтримкою бази данних програмного забезпечення на кожному потоці. Конфігурація проводиться програмним забезпеченням, а розміри передачі між джерелом та пунктом призначення не залежать[14].

Від Інтерфейсу в інтегральній схемі (I²C) аж до трьох інтерфейсів шини I2C можуть працювати в режимі багатомастерного. Воони уможливлюють підтримку стандартного та швидкого режиму. Можлива підтримка ними 7/10-бітного режиму адресації та 7-бітного режиму подвійної адресації (як підлеглий). Вбудована апаратна перевірка та CRC-генерація . Їх може обслуговувати DMA і вони підтримують SMBus 2.0 також PMBus.

Послідовний периферійний інтерфейс (SPI) BF537 оснащений до трьох SPI в підрядному та ведучому режимі, у режимах повного дуплексу та симплексного зв'язку. SPI1 може розганяти швидкість комунікації до швидкості 30 Мбіт / с., тоді як SPI2 та SPI3 можуть комунікувати зі швидкістю до 15 Мбіт/с з 3-х бітним домасштабним що дає 8 головних частотних режимів, а кадр може налаштовуватися на 8 або 16 біт. Генерація / верифікація апаратної CRC підтримує основні режими SD Card / MMC. Всі SPI обслуговуваються також за допомогою DMA-контролера. Інтерфейс SPI може також налаштовуватись для роботи в режимі TI для зв'язку в головному і підрядному режимі[14].

Висновок: Було аналізовано та досліджено компоненти, які будуть брати участь у побудові телеметричної системи відео моніторингу складних природніх об’єктів.

## 2 Вибір та обґрунтування схемотехнічного рішення

У якості теплівізійного пристрою було обрано використовувати тепловізійний пристрій MicroCAM 3. Цей тепловізійний пристрій увібрав у себе потрібні функції для системи. Ядро MicroCAM має можливість цифрового виходу з 60-ти вихідного роз'єму на ядрі MicroCAM 3. Пристрій підтримує режим поступового збільшення та має опцію зберігання зображень та завантаження . До 32 зображень можуть бути зроблені та збережені у флеш-пам'яті. Ця функція підтримується командами RS232[15].

Функції далекоміра буде виконувати модуль LW1012MU. За відносно невелику гроші, модуль пропонує велику дістанцію вимірювання у 1500м. Датчиком температури буде слугувати DS18B20[11]. Крім низької ціни він має наступні характеристики:

- Напруга живлення : 3v-5.5v;
- Протокол обміну даними: 1 - Wire;
- Спосіб підключення : прямій / по одній лінії з паразитним живленням;
- Дозвіл перетворення температури : 9 біт - 12 біт;
- Діапазон виміру температури : від - 55 до 125 oC;
- Період виміру температури при максимальній точності 12 біт: 750 мС;
- Тип індексації на лінії 1 - Wire: унікальна 64-бітова адреса;

Було вирішено використовувати у якості мікроконтролера модель ADSP - BF537, яка є високоінтегрованими рішеннями для покоління вбудованих застосувань, підключених до мережі. Завдяки поєднанню стандартних інтерфейсів з високопродуктивним ядром для обробки сигналів можна швидко розробити економічно ефективні застосування без необхідності використання дорогих зовнішніх компонентів[16]

Висновок: Було аналізовано та обрано компоненти, які у взаємодії покажуть хороший результат.

## 3. Розробка структурної схеми системи

Система тепловізійного прицілу призначена для розширення функціональних можливостей тепловізійного пристрою у нічний період доби. Принцип роботи системи заснований на вимірюванні вхідних даних, обробка вхідних даних у реальному часі та подання результату на дисплей.

Структурна схема( Рис 3.1) системи тепловізійного прицілу наведена нижче

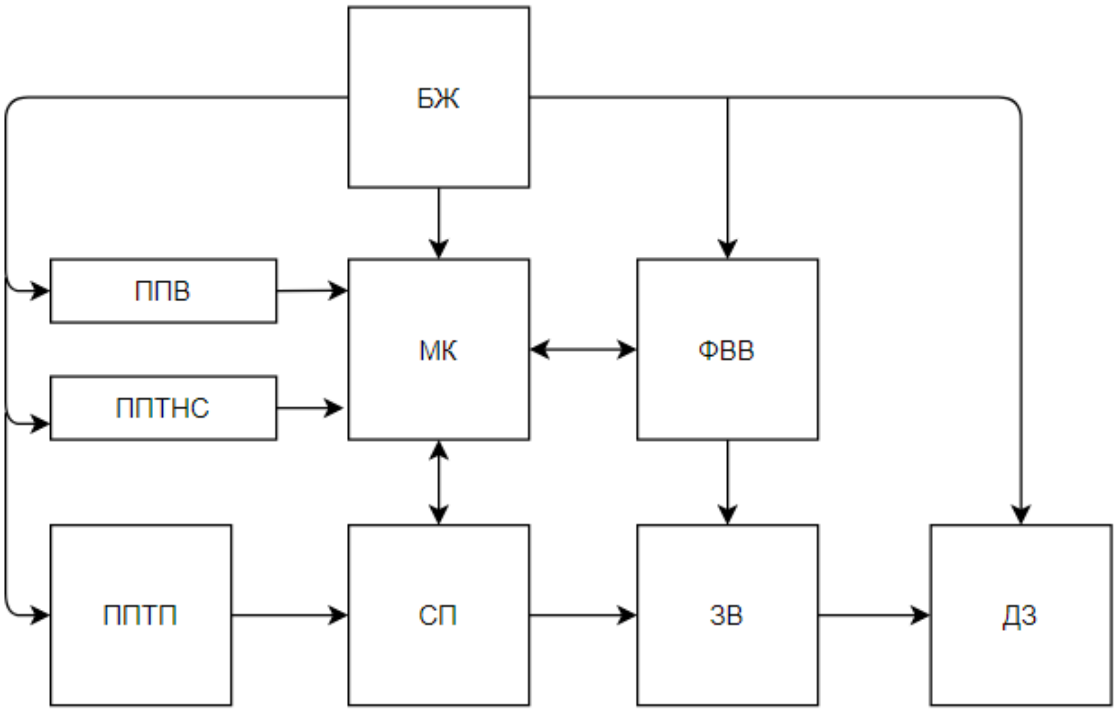


Рисунок 3.1 – структурна схема системи тепловізійного прицілу

На схемі позначено:

ППТНС – первинний перетворювач температури навк. середовища

ППВ – первинний перетворювач відстані

ППТП– первинний перетворювач теплового поля

СП – сигнальний процесор відеопотоку

МК – мікроконтролер

ЗВ – змішувач відеопотоків

ВФФ – формувач вторинного відеопотоку

ДЗ – дисплей відеозображення

БЖ – блок живлення.

Комплекс ,побудовано за модульним принципом. Блок ППТП складається с головки тепловізійного пристрою. Його функція формування растрового зображення із навколишнього середовища для подальшого його перетворення у сигнальному процесорі.

Сигнальний процесор призначений для перетворення растрового зображення у відеопотік у реальному часі. Висока швидкість перетворення потоку досягається за допомогою високошвидкісних апаратних схем.

Первинним перетворювачем відстані слугує модуль лазерного далекоміра, який під’єднаний через інтерфейс RS-232 до мікроконтролера. Він формує цифрові сигнали на основі часу, за який промінь лазера дістався цілі і повернувся назад.

Первинним перетворювачем температури навколишнього середовища слугує датчик температури. Його головна функція перетворення температури датчика у цифровий код.

Блоком МК слугує мікроконтролер,  який через RS-232 отримує значення з первинних перетворювачів. Далі обробляє дані та формує статичне зображення у вигляді цифрового коду. Потім за допомогою спеціально розробленого алгоритму на основі балістичних таблиць  для конкретного калібру кулі, розраховується координати прицілу правильної траєкторії руху кулі. В результаті прицільна сітка зміщується відносно координат центру в залежності від визначених координат траєкторії.

У блоці ВФФ формується відеопотік у вигляді прицільної сітки та тексту, на основі результатів обробки координат мікроконтролером.

Блок ЗВ служить для змішування відеопотоків з блоків СП та ФВВ. Тобто йде процес накладання відеопотоку, який сформує тепловізійна голівка, з динамічним відеопотоком вторинного формувача.

Блок ДЗ відображує змішане зображення користувачу на дисплеї.Для живлення всіх блоків системи служить блок живлення. Його функція полягає в перетворенні напруги батарей акумуляторів в напруги з рівнями, необхідними для роботи всіх блоків системи.

Висновок: Було розроблена структурна схема, на основі якої можна починати розробляти функційну схему системи.

## 4. Розробка функційної схеми системи

Функційна схема системи тепловізійного прицілу розроблена на підставі стуктурної схеми. Тепловізійний пристрій складається з тепловізійної голівки, сигнального процесору на відповідно дисплею. Тому ідея системи в наступному, під’єднати між сигнальним процесором та дисплеєм наш прилад, який буде формувати відеопотік та накладати на відеопотік з сигнального процесору. Результат виводиться на відеодисплей. Розглянемо більше детально кожний із процесів. Узагальнена функційна схема представлена на рисунку 4.1.

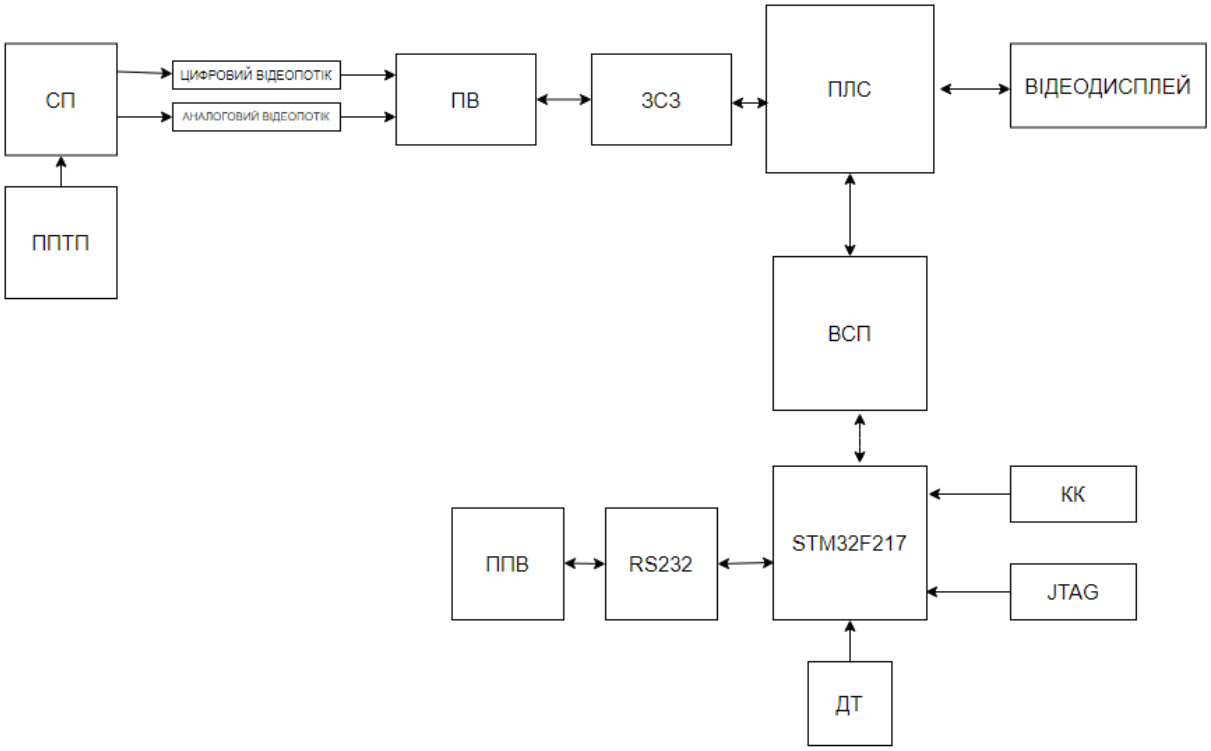


Рисунок 4.1 – Функційна схема

На схемі позначено:

СП сигнальний процесор

ВСП вторинний сигнальний процесор

ЗСЗ змішувач статичного зображення

ПВ перетворювач відеоформатів

МК мікроконтролер

ППТП первинний перетворювач теплового поля

ПЛС програмуюча логічна схема

КК кнопки користувача

ДТ датчик температури

ППВ первинний перетворювач відстані

### 4.1 Первинний перетворювач теплового поля

Розглянемо первинний перетворювач теплового поля. Як було зазначено на струткурній схемі, перетворювач буде формувати растрове зображення. Ним буде слугувати тепловізійна голівка MicroCam3(Рис 4.2).

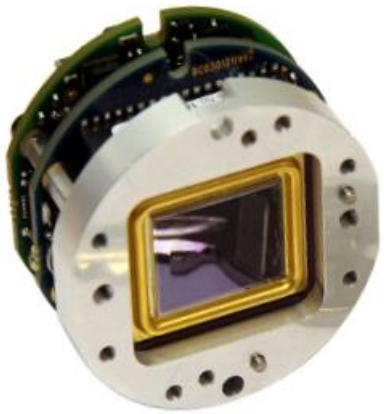

Рисунок 4.2 - MicroCam3

Ядро MicroCAM 3 з версією вбудованого ПО3х12, які оснащені базовими платами V5.8 і вище підтримують швидкий послідовний зв'язок через порт USB[15]. Сигнали USB спрямовуються з роз'єму камери на роз'єм Mini USB в інтерфейс MicroCam3. Тепловізійний пристрій підтримує функцію зберігання і завантаження зображень. До 32 зображень можуть бути захоплені і збережені у флєш-пам'яті камери. Ці зображення можуть бути викликані та завантаженні. Ця фунцкія підтримуються командами RS-232. Це особливо корисно для швидкого завантаження зображень[15].

Растрове зображення направляється по інтерфейсу RS232 до блоку сигнального процесору . Розглянемо більш детальніше інтерфейс RS232. Це стандарт фізичного рівня для асинхронного інтерфейсу UART, який призначений для організації прийому-передачі даних між передавачем або терміналом і приймачем або комунікаційним устаткуванням за схемою точка-точка. RS 232 є інтерфейсом для передачі інформації між двома пристроями на відстань до 20 метрів. Інформація передається по сполучних дротах з більш високою напругою, ніж

стандартна напруга 5 В, для більшої стійкості. Передача даних є асинхронною з близько встановленою швидкістю передачі[16].

### 4.2 Сигнальний процесор

Основна функція  блоку СП оцифровування зображення  , тобто перетворення зображення на відеопотік. Було вирішено використовувати універсальний тепловізійний модуль XCore серії LA(Рис-4.3). Сигнали USB спрямовуються з роз'єму камери на роз'єм Mini USB в інтерфейс MicroCam3.  Інтерфейсом слугує конектор Hirose з 60 пінами. Він включає в собі інтерфейс джерела живлення, інтерфейс послідовного зв'язку RS - 232, UART інтерфейс зв'язку, аналоговий відеоінтерфейс та цифровий вихід BT.656. Усі цифрові виходи за умовчанням відключені. Користувач може увімкнути його за допомогою програмного забезпечення контроллера GUI. Тільки один тип цифрового виходу підтримується одночасно[18].

Рисунок 4.3 - Тепловізійний модуль XCore серії LA

### 4.3 Перетворювач форматів

Оцифрований відеопотік посилається у блок ПВ, який перетворює відеопотік у  формат BT.656. Було вирішено використовувати універсальний одинчіповий мультиформатний відеодекодер ADV7180, який автоматично виявляє і перетворює PAL, NTSC і SECAM стандарти у вигляді композитного, S - Video і компонентного відео у цифровий формат BT.656. Вихід BT.656 працює з відеоданими формату Y'CbCr. Аналогові дані компонентного сигналу Y'PrPb у форматі 4: 2: 2

перетворюються у цифрові дані Y'CbCr, які потім мультиплексуються у потік 8 чи 10-розрядною шини у наступному порядку Cb0Y0Cr0Y1Cb2Y2Cr2 і так далі. Замість звичайних сигналів рядкової і кадрової синхронізації (HSYNC, VSYNC), які передавались в аналогових інтерфейсах, в BT.656 застосовуються унікальні тимчасові кодові повідомлення, вставлені у потік даних[19].

Після декодування відеопотоку приймач повинний визначити кадрову мі – тку і по відношенню до неї виконати "нарізку" послідовного потоку на слова і відновити паралельний потік даних. Цей процес називається фреймуванням. Для того щоб сформувати структуру кадру для потоку, треба використати унікальне синхрослово, яке б не зустрічалося у полі даних зображення. Це синхрослово буде періодично передаватися у послідовному потоці, щоб створити опорну тимчасову мітку, по відношенню до якої і почнеться складання 8, - або ж 10-разрядних слів інтерфейсу[19]. Усі цифрові відеоформати, підтримувані SDI - оболонкою, використовують один і той же опорний синхросигнал TRS (Timing Referance Symbol). Цей символ передається під час неактивного відео. Існують два типи TRS - сигналу, які передаються для кожного рядка кадру : перший передається на початку зони сигналу активного відео SAV, інший - у кінці зони активного відео EAV. Синхросимвол TRS складається з чотирьох послідовних слів:

3ff 000 000 XYZ (4.4)

Перше слово у шістнадцятирічному коді - 3ff, а останнє слово - XYZ. Перші три слова TRS незмінні і називаються преамбулою і представляють унікальну бітову комбінацію потоку. Четверте слово, назване XYZ, залежить від типу специфічного цифрового потоку відео. Для визначення меж прийнятого кадру дешифратор фрейму повинен спочатку отримати підряд 10 одиниць, а потім 20 підряд наступних сигналів Лог. "0"[19].

Аналоговий інтерфейс ADV7180 містить швидкісний 10-бітний аналого-цифровий перетворювач (АЦП), який оцифровує аналоговий відеосигнал. Аналоговий інтерфейс використовує диференціальні канали для АЦП для забезпечення високої продуктивності в додатках зі змішаним сигналом. Зовнішній інтерфейс також включає 3-канальний вхідний мультиплексор, який дозволяє оброблювати

декілька композитних відеосигналів. Струмові затиски розташовані перед АЦП, щоб забезпечити залишання відеосигнала в межах діапазону перетворювача. Точна фіксація відеосигналу виконується вниз за течією за допомогою цифрового точного затиску всередині. Далі, відформатований відеопотік направляється до вторинного сигнального процесора, через адаптер USB/VGA[20].

У більшості мікросхем декодерів і відеопроцесорів передбачено використання декількох типів аналогових відеоінтерфейсів. Для забезпечення аналого-цифрового перетворення можуть використовуватися від 2 до 3 вбудованих швидкісних АЦП. Розрядність АЦП 8 або 10 розрядів. Для перетворення компонентного і RGB- відеосигналу використовується 3-канальний АЦП. Структура конвертора містить також необхідні схеми прив'язки рівня чорного по кожному каналу і нормуючому підсилювачу. Виділення сигналів синхронізації з композитного сигналу забезпечує вбудований модуль селектора синхроімпульсів. У декодерах є вбудований модуль розділення сигналів яскравості і різнокольорових сигналів. Часто ці модулі називають процесорами, оскільки окрім розділення сигналів на компоненти здійснюється додаткова цифрова обробка з метою усунення артефактів декодування і поліпшення якості сигналів[19]. При використанні вхідного сигналу у вигляді композитного сигналу, перетворення в цифрову форму робиться за наступною схемою:

1) Фільтрація колірних компонентів і сигналу яскравості.
2) Аналого-цифрове перетворення трьох компонентів з частотою 27 МГц.
3) Декодування компонентів (формат 4: 4: 4).
4) Перетворення сигналів Y'Cb, Cr у формат 4: 2: 2, BT.656).
5) Транспонування діапазонів сигналів Y, Cr, Cb відповідно до BT.656;
6) Вставка кодів синхронізації SAV і EAV.
7) Перетворення в послідовний код

Остання процедура може застосовуватися тільки при передачі оцифрованого відеосигналу у послідовному форматі.

У деяких цифрових відеосистемах етап (1) пропускається. Композитний сигнал перетворюється у цифровий формат, а потім вже в цифровій формі робиться

виділення сигналу яскравості. Проте аналогова фільтрація дає кращу якість і апаратно реалізується простіше[19].

### 4.4 Перетвоювачі та обробка мікроконтролера

У той же час мікроконтролер починає формувати вторинне зображення у вигляді відформатованої прицільної сітки, на основі отриманих даних з блоків ППВ і ДТ. Первинним перетворювачем відстані слугує модуль далекоміра LW1012MU(Рис 4.5). Його максимальна дістанція, яку він може виміряти 1500 метрів, при робочій довжині хвилі 905 нм. Достатньо точний, адже похибка +/- 0,5 метрів[21].

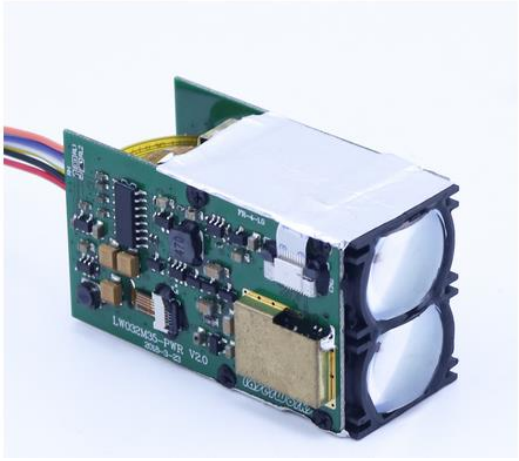

Рис 4.5- модуль далекоміра

DS18B20(Рис 4.6) слугує датчиком температури. Він перетворює температуру у цифрові дані, які зберігаються у 2 байтах регістру. Через регістр конфігурації можна встановити розширення перетворення термодатчика, який може бути заданий 9, 10, 11 або 12 біт. Регістр конфігурації і пороги аварійного сигналу містяться в енергонезалежній пам'яті (EEPROM)[11]. Для обміну даними використовується інтерфейс 1-Wire. Це протокол передачі даних по одному дроту в обидві сторони. Має асинхронний та полудуплексний режим зв’язку. Принцип побудований на тому, що у нас завжди є тільки один головний пристрій на шині, який посилає команди, і дочірні пристрої, які ці команди приймають та відповідають на них.

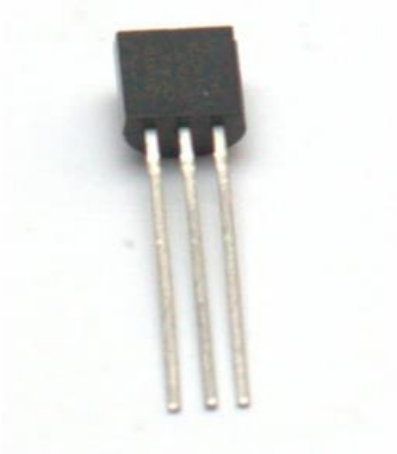

Рис 4.6 - DS18B20

За швидкість та надійність функціонування системи відповідає мікроконтролер BF537, який забезпечує продуктивність до 600 МГц. Блок-схема позначена на рисунку 4.7. Він отримує значення через RS232 та 1-Wire з первинних перетворювачів. У результаті мікроконтролер формує зображення прицільної сітки, із зміщеними координатами прицілу. Зміщення координат прицілу відбувається за рахунок алгоритму балістичних таблиць. Тепер поговорим більш детально про сам мікроконтролер. ADSP -BF524 є членом сімейства продуктів Blackfin, в яких використовується архітектура аналогових облаштувань[17]. Процесори Blackfin® поєднують в собі сучасний процесор обробки сигналів з двома MAC, переваги чистої, ортогонального RISC- подібного набору команд мікроконтролера і мультимедійні можливості з однією командою і декількома даними (SIMD).

Щоб запустити процесор на повну продуктивність, прискорювач реалізує чергу попереднього вибору інструкцій та кеш гілки, що збільшує швидкість виконання програми із 128-бітною флеш-пам’яттю[17].

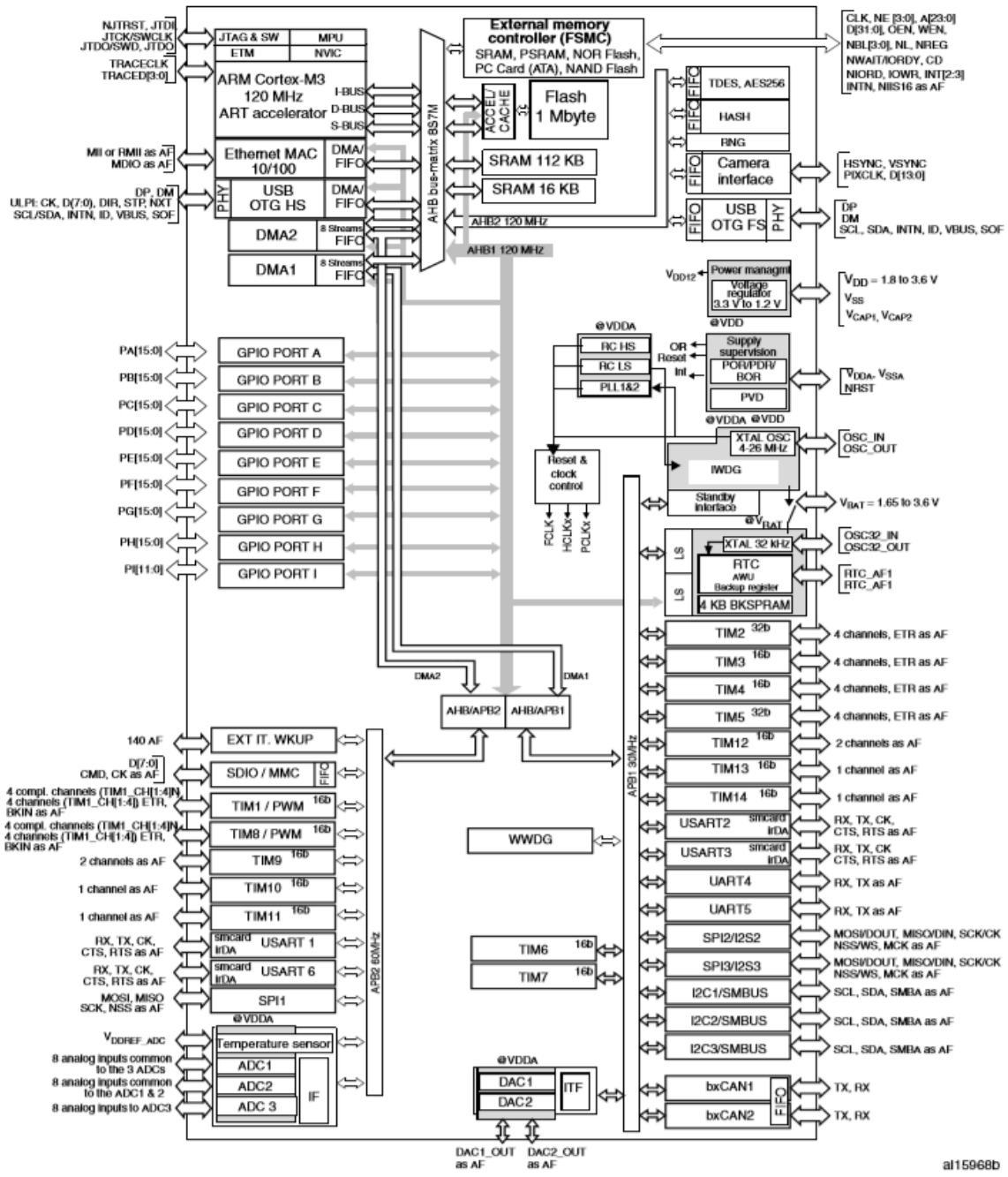


Рисунок 4.7 блок-схема мікроконтролера

Блок захисту пам'яті (MPU) використовується для управління доступом CPU до пам'яті, щоб запобігти випадковому пошкодженню пам'яті або ресурсів, використовуваних будь-яким іншим активним завданням. Ця зона пам’яті організована до 8 захищених відділів, які в свою чергу можна розділити на 8 підвідділи. Розміри зони захисту - від 32 байтів і цілих 4 гігабайти адреси.MPU особливо

корисний для програм, де певний критичний або сертифікований код має бути захищений від непередбачливої поведінки інших завдань[17].

У мікроконтролер вбудовано 128-бітну флеш-пам’ять. На пристроях також є 512 байтна пам'ять OTP, яку можна використовувати для зберігання важливих користувацьких даних.

Пристрій має два DMA, тобто два прямих доступи до пам'яті з загальним призначенням (DMA1 та DMA2) з 8 потоками кожен. Вони здатні керувати передачею з одної пам'яті до іншої, периферійною пам’яттю та переходом пам’яті до периферії. Вони поділяють деякі централізовані FIFO тобто алгоритм заміщення комірок пам'яті для периферійних приладів APB (перифирійна шина)/ AHB (вдосконалена високопродуктивна шина), підтримують передачу пакетних передач і призначені для забезпечення максимальної периферійної пропускної здатності для (AHB / APB)[17].

Два контролери DMA підтримують кругове управління буфером, так що конкретний код не потрібен, коли контролер доходить до кінця буфера. Два контролери DMA також мають функцію подвійної буферизації, яка створює автоматизування та об'єднання двох буферів пам'яті, не вимагаючи спеціального коду для цього.

Кожен потік підключений до апаратних запитів DMA з підтримкою бази данних програмного забезпечення на кожному потоці. Конфігурація проводиться програмним забезпеченням, а розміри передачі між джерелом та пунктом призначення не залежать. DMA можна використовувати з основними периферійними пристроями:

- SPI та I2S
- I2C
- USART та UART
- ЦАП
- Інтерфейс камери (DCMI)
- АЦП.

Між інтегрованим звуком (I2S) Доступні два стандартні інтерфейси I2S (мультиплексовані з SPI2 та SPI3). Вони можуть працювати в головному або підрядному режимі, в режимах симплексного зв'язку і можуть бути налаштовані на роботу з 16- / 32-бітною роздільною здатністю як канали вводу або виводу. Підтримуються частоти відбору звуку від 8 кГц до 192 кГц. Коли будь-який або обидва інтерфейси I2S налаштовані в головному режимі, головний тактовий годинник може бути виведений на зовнішній ЦАП / КОДЕК у 256 кратній частоті вибірки. Всі інтерфейси I2Sx можуть знову ж таки обслуговуватися DMA-контролером[17].

Пристрої мають додаткову, спеціально виділену PLL для аудіо I2S програми. Це дозволяє досягти точності тактового відбору часу а також відбір I2S без впливу на продуктивність процесора за використання USB-периферії. Конфігурацію PLLI2S можна змінити задля управління швидкісної зміни I2S без відключення основного PLL (PLL), що використовується для процесорів, USB та Ethernet інтерфейсів. Сформоване зображення передається по RS232 до вторинnго сигнального процесора, який оцифровує вторинее зображення на відепотік та змішує с первинним відепотоком[17].

### 4.5 Змішувач відеопотоків

ПЛС, ЗСЗ та ВСП об’єднує в собі сигнальний процесор AVR8003. Саме тут і йде формування нового відеопотоку та накладання на вже існуючий. Змішувач статичного зображення, накладає на головний відеопотік текст та ефекти, які було попередньо запрограмовано виробником AVR8003. Вторинний сигнальний процесор перетворює статичне зображення з мікроконтролера у вторинний відеопотік, у свою чергу програмуюча логічна система змішує два відеопотоки. Розглянемо вториний сигнальний процесор більш детально. Відеоприймач приймає вихідні сигнали передавача ADV7180. Трансивер витягає відео формату BT.656 для обробки перед повторною вставкою відео в ADV8003 через відео контакти для виведення через перетворювачі. Цей вхід не підтримує операції EDID або HDCP. Підтримка "картинка в картинці" (PiP) можлива при отриманні відеоданих на більш ніж один з відеовходів, таких як 48-бітовий порт пікселів і послідовний

відеоприймач. Гнучкі функції цифрового інтерфейсу включають наступне: 48-бітовий піксельний порт для загальних відеоданих, 24-бітовий піксельний порт для зовнішнього OSD, якщо внутрішнє OSD ADV8003 не використовується[22].

ADV8003 має гнучке цифрове ядро, яке забезпечує безліч різних конфігурацій поодиноких, подвійних і потрійних каналів обробки відео. Обробка відео може бути розміщена першою в ланцюжку сигналів, щоб гарантувати, що усі виходи обробляються з найвищою якістю.

ADV8003 має два інтерфейси HDMI, які підтримують усі формати HDTV до 4k × 2k. Кожен інтерфейс HDMI оснащений повнофункціональним майстром CEC і мікроконтролерами на кристалі з майстрами DDC I2C для виконання операцій HDCP і EDID. Також функціонал включає в себе підтримку зворотного аудіо-каналу, аудіо-інтерфейс HDMI і підтримка декількох аудіо-форматів (S / PDIF, I2S, DSD)[22] .

ADV8003 оснащений високошвидкісним цифро-аналоговим відеокодером. Шість 12-розрядних ЦАП відеосигналу забезпечують підтримку аналогових композитних, S - Video і компонентних (YPrPb / RGB) аналогових виходів у стандартному або високому розширені. Також можливо включити відеокодер ADV8003 для роботи в одночасних режимах, коли виводяться обидва формати: SD і ED / HD[22].

### 4.6 Відображення результату

У результаті змішаний відеопотік передається через інтерфейс HDMI до відеодисплею. Інтерфейс HDMI, працює з використанням технології диференціальної сигналізації з мінімізацією переходів для передачі інформації або даних з одного місця в інше. Диференціальна сигналізація з мінімізацією переходів (TDMS) - це метод, який захищає інформацію від погіршення якості при переміщенні по довжині кабелю від одного пристрою до іншого[23]. Відправляючий пристрій кодує сигнал, зменшуючи кількість переходів. Це допомагає захистити якість сигналу і обмежує вірогідність погіршення якості. Коли інформація передається, один з витої пари кабелів несе сам сигнал, тоді як інший несе зворотну копію сигналу,

який знаходиться в передачі. Після прибуття на приймальне облаштування HDMI вимірює різницю між цими сигналами і використовує інформацію для компенсації втрати сигналу. HDMI ідеально підходить до нашої системи, адже це повністю цифровий кабель для передачі сигналу. Він несе нестислі, повністю цифрові дані, що передаються між різними компонентами. HDMI підтримує ідеальне перенесення зображень на різних етапах обробки, які існують в складних цифрових до аналогових або аналогових цифрових процесів. HDMI поставляється з одним кабелем зручності. Він пропонує однакову цифрову передачу для аудіосигналів за допомогою до восьми одночасних каналів аудіо високій чіткості[23].

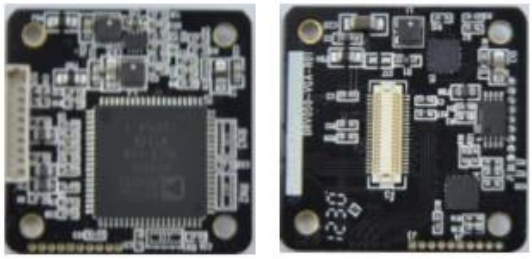

Рисунок 4.7- плата управління DRV050-VGA-R02

Було вирішено використовувати відеодисплей під назвою SVGA Micro – OLED, з розширенням SVGA (800×600), з платою управління DRV050-VGA-R02(Рис 4.7). Декодер може автоматично виявляти і перетворювати графічні сигнали RGB в цифрові Компонентні відеодані RGB 4 : 4: 4, сумісні з 8-бітовим стандартом ITU - R BT.656. Кремнієва основа зроблена, користуючись технологією CMOS, яка забезпечує цифрову відеообробку і розширення 800 (× 3) × 600. За рахунок двохпроводного послідовного інтерфейсу звязку можна регулювати яскравість, контраст, колір та корегування гамми. Зміна значень регістру декодера і дисплея набуде чинності негайно, але при відключенні живлення конфігурація буде втрачена[24].

Розглянемо приклади команд. Кожна команда повинна бути відправлена за 600 мс, а загальна кількість байтів має бути менше 64 байт, інакше ми отримаєм помилку.

1. Змінити контраст дисплею 02 21 03 09 XX 03(XX це значення, яке ми хочемо встановити, діапазон від 00h до FFH)
2. Змінити  яскравість дисплею 02 24 03 00 XX 03
3. Reset 02 55 03 00 00 03  та інші.

Висновок: Було аналізовано та розроблена функційна схема, на основі структурної схеми. Також було розглянуто кожен із компонентів схеми більш детально.

## 5. Опис алгоритму роботи системи

Ідея системи полягає у тому, що користувачеві дається змога значно спростити процес попадання у ціль, шляхом автоматичного знаходження правильної траєкторії пострілу у ціль, та відповідно відображення її на дисплеї з тепловізійним зображенням. Якщо у цілому, то алгоритм роботи телеметричної системи відеомоніторингу складних природніх об’єктів, полягає у змішувані двох відепотоків та відповідно відображення результату на відеодисплеї. Розглянемо більш детально кожен із процесів.

Тепловізійний пристрій виступає у ролі тепловізійної голівки. Її функція у формуванні тепловізійного растрового зображення, яке у результаті оцифровується сигнальним процесором. Тобто, звичайне растрове зображення з тепловізійної голівки перетворється в аналоговий відеопотік. Але для наступних перетворень, відеопотік повинен бути у цифровому форматі, адже ми не можемо напряму подати його на змішувач сигналів, попередньо не перетворивши. Тому було вирішено використовувати перетворювач відеоформатів ADV7180.

Функція перетворювача форматів полягає у перетворені формату відеопотоку на формат композитного відеосигналу CVBS. Після перетворення , відформатований відеопотік направляється до плати ADV8003.Також за допомогою інтерфейсів USB/VGA, які з’єднані між собою відповідним адаптером, мікроконтролер може приймати участь в управлінні роботи схеми ADV7180, а саме у

змінні відеформату перетворення, за допомогою спеціальний бібліотечних команд.

У той же час, мікроконтролер формує статичне зображення у вигляді тексту та прицільної сітки. На основі даних з первинних перетворювачів, а саме дальності до об’єкта та температури навколишнього середовища, він вираховує правильну траєкторії до цілі на основі балістичних таблиць. В результаті відбувається зміщення координат прицілу відносно центру, для надання правильної траєкторії кулі, та заповнення поля для тексу значенням відстані та температури. Далі мікроконтролер передає по COM порту оброблені дані, та починає регулювання процесу змішування.

Схема ADV8003 виконує одразу декілька функцій. Спочатку вона перетворює данні з мікроконтролера на відепотік, який у результаті змішує з первинним відформатованим відеопотоком з тепловізійної голівки. З рештою отриманий відеопотік, перетворюється на цифровий формат HDMI та передається через інтерфейс HDMI на відеодисплей. У результаті користувач отримує тепловізійний відеопотік, з накладеним зображенням у вигляді значення дальності до цілі, температури навколишнього середовища та відформатованих координат прицільної сітки. За допомогою цих параметрів та враховуючи похибки, траєкторія кулі стане правильною, що дозволить користувачеві успішно влучити у ціль.

Висновок: Було розглянуто та аналізовано алгоритм роботи системи.

## 6. Розробка блок-схеми програмного забезпечення системи

Система тепловізійного прицілу, для змішування відеопотків, використовує власне розроблене програмне забезпечення. Блок-схема програмного забезпечення системи наведена на рисунку 6.1.

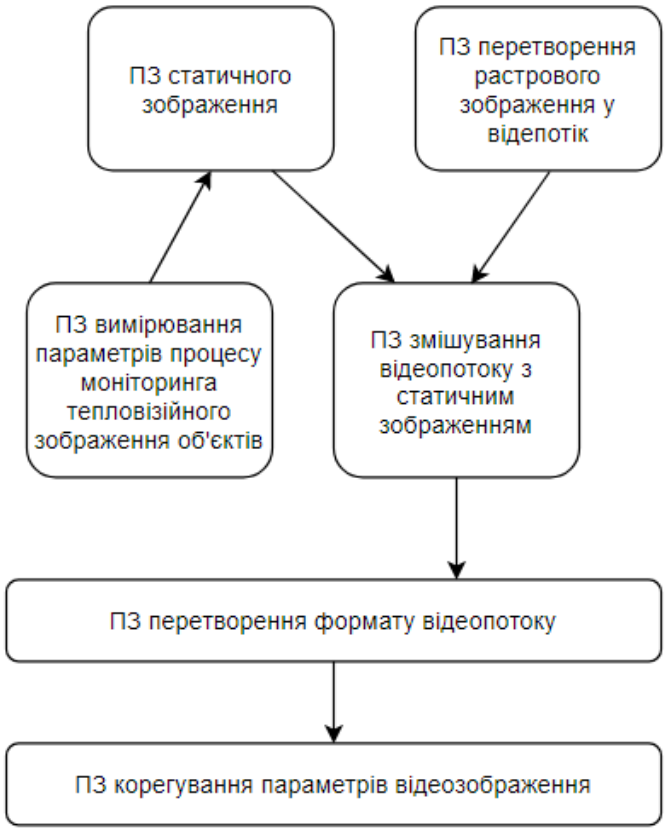


Рисунок 6.1 – Блок-схема системи тепловізійного прицілу

ПЗ перетворення растрового зображення у відеопотік формує первинний відеопотік. Це відбувається за рахунок того, що тепловізійна голівка формує растрове зображення. Сигнальний процесор у свою чергу, формує з цього зображення відеопотік у відповідному відеоформаті для подальшого його перетворення.

Одночасно з перетворенням растрового зображення, ПЗ вимірювання параметрів процесу моніторинга тепловізійного зображення об’єктів, виконує функцію вимірювання та збору даних про об’єкт та навколишнє середовище. А саме, вимірює  дальність до об’єкту та температуру навколишнього середовища. Саме

за допомогою цих даних та алгоритму балістичної таблиці і йде розрахування та формування відповідних координат прицілу.

ПЗ статичного зображення формує відповідно статичне зображення, у якому відображує результат ПЗ вимірювання параметрів. Зображення передається у вигляді тексту та прицільної сітки. Текст відповідно виводиться у вигляді значення дальності до об’єкту або температури навколишнього середовища. Прицільна сітка формується на основі результату з ПЗ вимірювання параметрів, а саме відбувається зміщення координат прицілу відносно центру сітки, залежачи від відстані та температури.

ПЗ змішування відеопотоку з статичним зображенням виконує змішуючу функцію. А саме, змішує отриманий оцифрований відеопотік з статичним зображенням. У результаті виходить щось на кшталт субтитрів.

Дуже важливим процесом є перетворення відеоформатів. Адже існує велика кількість відеоформатів і для коректного відображення відепотоків повинен бути один і той же відеоформат. Саме тому ПЗ перетворення формату відеопотоку і присутнє у нашій блок-схемі. Наприклад, перетворення з цифрового формату у аналоговий та навпаки.

В кінці, отриманий результат проходить обробку від ПЗ корегування параметрів відеозображення. А саме він формує параметри зображення, тобто регулює контрастність, інверсію кольору, від чорно-білого до біло-чорного. Також можна регулювати розмір зображення. А головне, для більш зручної взаємодії з пристроєм існують кнопки користувача. Їх функція полягає у ручному переміщенні та регулюванні накладеним зображенням на дисплеї. Наприклад, якщо користувач не хоче бачити значення дальності до об’єкта або він хоче щоб було зручно бачити сам відеопотік, він може за допомогою натиснення кнопок користувача, змістити текст наліво, направо, вверх чи вгору відносно центру.

Висновок: Було аналізована та розроблена блок-схема програмного забезпечення системи.

## 7 Розробка принципової схеми

У даному розділі дипломного проекту буде розглянуто кожен із компонентів системи тепловізійного пристрою на рінві мікромсхем.

### 7.1 Мікроконтролер

Почнемо з мікроконтролера системи, функція якого у формуванні статичного зображення у вигляді тексту та керуванням процесу змішування відеопотоків. Принципова схема приведена на рисунку 7.1.

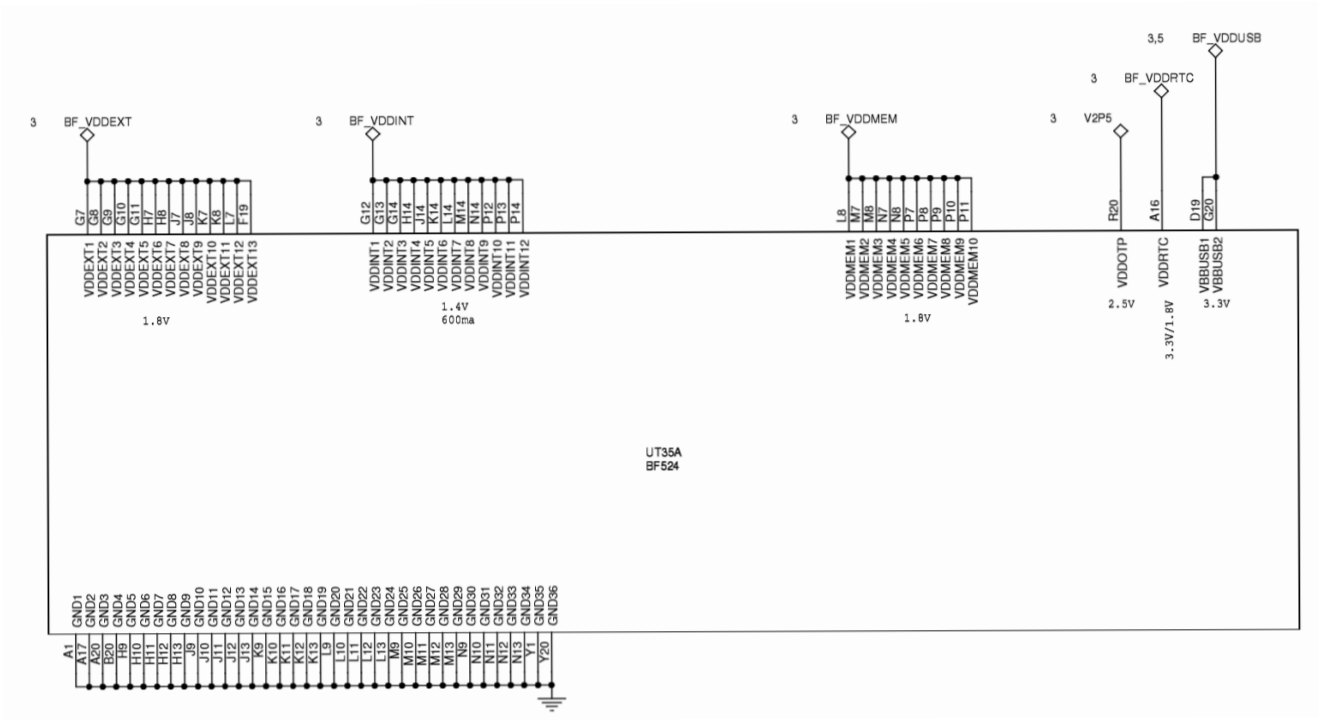


Рисунок 7.1 – Принципова схема мікроконтролера ADSP-BF524

Мікроконтролери ADSP - BF537 є високоінтегрованими рішеннями для покоління вбудованих застосувань, підключених до мережі. Завдяки поєднанню стандартних інтерфейсів з високопродуктивним ядром для обробки сигналів можна швидко розробити економічно ефективні застосування без необхідності використання дорогих зовнішніх компонентів[17]. Системні периферійні пристрої включають IEEE- сумісний 802.3 10/100 Ethernet MAC, високошвидкісний

OTG- контроллер USB 2.0, TWI- контроллер, флэш-контроллер NAND, два порти UART, порт SPI, два послідовні порти (SPORT), вісім універсального призначення 32-розрядні таймерів з можливістю ШИМ, основним таймером, годинником реального часу, сторожовим таймером, інтерфейсом Host DMA (HOSTDP) і паралельниі периферійні інтерфейси (PPI)[17].

Мікроконтролери ADSP - BF537 містять багатий набір периферійних пристроїв, підключених до ядра через декілька шин з високою пропускною спроможністю, що забезпечує гнучкість конфігурації системи і відмінну загальну продуктивність системи. Ці процесори Blackfin містять виділені модулі мережевого зв'язку і високошвидкісні послідовні і паралельні порти, контроллер переривань для гнучкого управління перериваннями від вбудованих периферійних пристроїв або зовнішніх джерел, а також функції управління управлінням живленням для налаштування продуктивності і характеристик потужності процесора і системи для багатьох прикладних сценаріїв. Усі периферійні пристрої, за винятком універсального введення-виведення, TWI, годинника реального часу і таймерів, підтримуються гнучкою структурою DMA. Існують також окремі канали пам'яті DMA, призначені для передачі даних між різними областями пам'яті процесора, включаючи зовнішню SDRAM і асинхронну пам'ять[17].

Декілька вбудованих в кристал шин з частотою до 133 Мгц забезпечують достатню смугу пропускання для підтримки працездатності ядра процесора і активності на усіх вбудованих і зовнішніх периферійних пристроях. Процесор включає чіповий регулятор напруги для підтримки динамічного управління живленням процесора.

Регулятор напруги в діапазоні рівнів напруги при живленні від VDDEXT. Регулятор напруги можна обійти на розсуд користувача[17].

### 7.1.1 Ядро мікроконтролера

Ядро процесора Blackfin містить два 16-розрядні помножувачі, два 40-розрядні акумулятори, два 40-розрядних АЛП, чотири відео АЛП і 40-розрядний

сдвигатель. Обчислювальні блоки обробляють 8 -, 16 - або 32-бітові дані з файлу регістрів. АЛП виконують традиційний набір арифметичних і логічних операцій з 16-бітовими або 32-бітовими даними. Крім того, багато спеціальних інструкцій включені для прискорення різних завдань обробки сигналів. Вони включають бітові операції, такі як витягання поля і підрахунок населення, множення по модулю 232, ділення примітивів, насичення і округлення, а також виявлення знаку / експоненти. Набір відеоінструкцій включає операції вирівнювання і упаковки байтів, 16-бітові і 8-бітові операції додавання з відсіченням, 8-бітові середні операції і 8-бітові операції віднімання / абсолютного значення / накопичення (SAA). Також передбачені інструкції порівняння / вибору і пошуку вектору. Для деяких інструкцій дві 16-бітові операції АЛП можуть виконуватися одночасно для пар регістрів (16-бітова старша половина і 16-бітова молодша половина обчислювального регістра). Якщо використовується другий АЛП, можливі четверні 16-бітові операції[17].

Файл обчислювальних регістрів містить вісім 32-бітових регістрів. При виконанні обчислювальних операцій з даними 16-бітових операндів файл регістрів працює як 16 незалежних 16-бітових регістрів. Усі операнди для обчислювальних операцій поступають з мультипортированного файлу регістрів і полів констант команд. Кожен MAC може виконувати 16-бітове 16-бітове множення в кожному циклі, накопичуючи результати в 40-бітових акумуляторах. Підтримуються підписані і непідписані формати, округлення та насиченість.

40-бітове облаштування зрушення може виконувати зрушення і обертання і використовується для підтримки інструкцій по нормалізації, витяганню полів і внесенню полів. Секвенсор програм контролює хід виконання команд, включаючи вирівнювання і декодування команд. Для управління потоком програм секвенсор підтримує відносні і непрямі умовні переходи ПК (із статичним пророцтвом переходів) і виклики підпрограм. Передбачено апаратне забезпечення для підтримки циклів з нульовим навантаженням. Архітектура повністю взаємозв'язана, що означає, що програмістові не треба управляти конвеєром при виконанні інструкцій із залежностями даних. Блок адресної арифметики надає дві адреси

для одночасних подвійних вибірок з пам'яті. Він містить мультипортированный регістровий файл, що складається з чотирьох наборів 32-бітового індексу, регістри модифікації, довжини і бази (для циклічної буферизації) і вісім додаткових регістрів 32-бітових покажчиків (для маніпулювання індексованим стеком в стилі C). Процесори Blackfin підтримують модифіковану архітектуру Гарварду у поєднанні з ієрархічною структурою пам'яті[17]. Пам'ять рівня 1 (L1) - це пам'ять, яка зазвичай працює на повній швидкості процесора практично без затримки. На рівні L1 пам'ять команд містить тільки інструкції. Дві пам'яті даних містять дані, а в спеціальній пам'яті даних блокнота зберігаються дані про стек і локальні змінні. Крім того, передбачені декілька блоків пам'яті L1, що пропонують поєднання SRAM і кеша, що настроюється. Блок управління пам'яттю (MMU) забезпечує захист пам'яті для окремих завдань, які можуть працювати на ядрі, і може захищати системні реєстри від неумисного доступу.

Набір команд процесора Blackfin оптимізований таким чином, що 16-бітові коди операцій представляють найчастіше використовувані інструкції, що призводить до чудової щільності скомпільованого коду. Складні інструкції DSP кодуються в 32-бітові коди операцій, що представляють повнофункціональні багатофункціональні інструкції. Процесори Blackfin підтримують обмежену багатозадачність, при якій 32-бітова інструкція може бути видана паралельно з двома 16-бітовими інструкціями, що дозволяє програмістові використати багато основних ресурсів в одному циклі команд. Складання процесора Blackfin мова використовує син-налог алгебри для простоти кодування і легкості для читання. Архітектура була оптимізована для використання разом з компілятором C / C++, що привело до швидкої і ефективної реалізації програмного забезпечення[17].

### 7.1.2 Архітектура пам’яті

Процесор Blackfin розглядає пам'ять як єдиний 4-байтовий адресний простір, використовуючи 32-бітові адреси. Усі ресурси, включаючи внутрішню пам'ять, зовнішню пам'ять і регістри управління введенням / виведенням, займають окремі розділи цього загального адресного простору. Частини пам'яті цього

адресного простору організовані в ієрархічну структуру, щоб забезпечити хороший баланс витрат і продуктивності деякої дуже швидкої внутрішньокристальної пам'яті з малою затримкою, такий як кеш-пам'ять або SRAM, і більшій, менш витратній і високопродуктивній внекристальной пам'яті системи. Див. Малюнок 3. Вбудована система пам'яті L1 - це високопродуктивна пам'ять, доступна процесору Blackfin. Система зовнішньої пам'яті, доступ до якої здійснюється через інтерфейсний модуль зовнішньої шини (EBIU), забезпечує розширення за допомогою SDRAM, флэш-памяти і SRAM, опционально дістаючи доступ до до 132 Мбайт фізичної пам'яті. Контроллер DMA пам'яті забезпечує переміщення даних з високою пропускною спроможністю. Він може виконувати блокову передачу коду або даних між внутрішньою пам'яттю і зовнішніми елементами пам'яті[17].

### 7.1.3 Одноразова програмована пам'ять

Процесор має 64k біт одноразової програмованої енергонезалежної пам'яті, яка може бути запрограмована розробником тільки один раз. Він включає масив і логіку для підтримки доступу для читання і програмування. Крім того, його сторінки можуть бути захищені від запису. OPT дозволяє розробникам зберігати як загальнодоступні, так і особисті дані на кристалі. На додаток до зберігання даних відкритого і закритого ключів для додатків, що вимагають безпеки, це також дозволяє розробникам зберігати повністю визначувані користувачем дані[17].

### 7.1.4 DMA контроллери

Процесор має декілька незалежних каналів DMA, які підтримують автоматичну передачу даних з мінімальними витратами для ядра процесора. Передачі DMA можуть відбуватися між внутрішньою пам'яттю процесора і будь-яким з його периферійних пристроїв з підтримкою DMA[17]. Крім того, передачі DMA можуть виконуватися між будь-якими периферійними пристроями з підтримкою DMA і зовнішніми пристроями, підключеними до інтерфейсів зовнішньої пам'яті, включаючи контроллер SDRAM і контроллер асинхронної пам'яті. До периферійних пристроїв з підтримкою DMA відносяться Ethernet MAC, NFC,

HOSTDP, USB, SPORT, порт SPI, UART і PPI. Кожен окремий периферійний пристрій з підтримкою DMA має щонайменше один виділений канал DMA. Контроллер DMA процесора підтримує як одновимірні (1 - D), так і двовимірні (2 - D) передачі DMA. Ініціалізація передачі DMA може бути реалізована з регістрів або з наборів параметрів, що називаються блоками дескрипторів. Можливість 2 - D DMA підтримує довільні розміри рядків і стовпців до 64k елементів по 64k елементів і довільні розміри кроків і стовпців до ± 32k елементів. Крім того, розмір кроку стовпця може бути менший, ніж розмір кроку рядка, що дозволяє реалізувати потоки даних, що чергуються[17]. Ця функція особливо корисна у відео додатках, де дані можуть чергуватися на льоту. Приклади типів DMA, підтримуваних процесором DMA, включають:

- Один лінійний буфер, який зупиняється після завершення,
- Круговий буфер автооновлення, який уривається в кожному повному або частково заповненому буфері.
- 1 - D або 2 - D DMA з використанням пов'язаного списку дескрипторів.
- 2 - D DMA з використанням масиву дескрипторів, що визначають тільки базову адресу DMA на загальній сторінці.

На додаток до виділених периферійних каналів DMA, для передачі між різні спогади про процесорну систему. Це дозволяє передавати блоки даних між будь-якою пам'яттю, включаючи зовнішню SDRAM, ROM, SRAM і флэш-память, з мінімальним втручанням процесора. Передачі DMA з пам'яті можуть контролюватися дуже гнучкою методологією на основі дескриптора або стандартним механізмом автобуфера на основі регістрів.

Процесор також має можливість зовнішнього контроллера DMA за допомогою двох зовнішніх виведень запиту DMA при використанні разом з інтерфейсним блоком зовнішньої шини (EBIU). Ця функціональність може використовуватися, коли високошвидкісний інтерфейс потрібно для зовнішніх FIFO і периферійних облаштувань зв'язку з високою пропускною спроможністю, таких як USB 2.0. Це дозволяє контролювати кількість передаваних даних в пам'ять DMA. Кількість передач на ребро програмованого. Ця функція може бути

запрограмована так, щоб пам'ять DMA мала підвищений пріоритет на зовнішній шині відносно ядра[17].

### 7.1.5 Порти UART

Процесори надають два повнодуплексних універсальних асинхронних порту приймача / передавача (UART), які повністю сумісні з UART стандарту ПК. Кожен порт UART надає спрощений інтерфейс UART для інших периферійних пристроїв або хостів, підтримуючи повнодуплексну асинхронну передачу послідовних даних з підтримкою DMA. Порт UART включає підтримку від п'яти до восьми бітів даних, одного або двох стопових бітів, і жодного, парного або непарного контролю парності. Кожен порт UART підтримує два режими роботи. Перший, PIO (запрограмоване введення / виведення) - процесор відправляє або отримує дані, записуючи або прочитуючи зіставлені регістри введення / виведення UART. Дані мають подвійну буферизацію як при передачі, так і при прийомі. Другий режим  DMA (прямий доступ до пам'яті) - контроллер DMA передає і ці передачі, і прийому. Це зменшує кількість і частоту переривань, необхідних для передачі даних в пам'ять і з неї. UART має два виділені канали DMA, один для передачі і один для прийому. Ці канали DMA мають нижчий пріоритет за умовчанням, чим більшість каналів DMA, із-за їх відносно низьких швидкостей обслуговування[17].

### 7.1.6 TWI інтрерфейс

Процесори включають модуль 2-дротяного інтерфейсу (TWI) для забезпечення простого методу обміну даними управління між декількома пристроями. TWI сумісний з широко використовуваним стандартом шини I2C®. Модуль TWI пропонує можливості одночасної роботи того, що веде і веденого, а також підтримку як 7-бітової адресації, так і арбітражу мультимедійних даних[17]. Інтерфейс TWI використовує два контакти для передачі тактового сигналу (SCL) і даних (SDA) і підтримує протокол зі швидкістю до 400 Кбит / с. Виведення інтерфейсу TWI сумісні з логічними рівнями 5 В. Крім того, модуль TWI повністю

сумісний з функціями шини управління послідовною камерою (SCCB), що спрощує управління різними сенсорними облаштуваннями камери CMOS[17].

### 7.2 SDRAM MT48H16M16

Наступний елемент SDRAM пам’ять, моделі Micron 256mb Mobile. Принципова схема наведена на рисунку 7.2. Micron SDRAM - це високошвидкісна КМОП-пам'ять з динамічним довільним доступом, що містить 268 435 456 біт. Він внутрішньо конфігурований як чотирьохканальний DRAM з синхронним інтерфейсом (усі сигнали реєструються на позитивному фронті тактового сигналу, CLK)[25]. Кожен з 67 108 864-битных банків x16 організований як 8 192 рядки по 512 стовпці по 16 біт. Кожен з 67,108,864-бітових банків x32 організований як 4096 рядків по 512 стовпці по 32 біта. Доступ до SDRAM для читання і запису орієнтований на пакетну обробку; Доступ розпочинається з вибраного місця розташування і триває впродовж запрограмованої кількості місць в запрограмованій послідовності. Доступ розпочинається з реєстрації команди ACTIVE, за якою йде команда READ або WRITE. Биті адреси, зареєстровані як співпадаючі з командою ACTIVE, використовуються для вибору банку і рядка, до якого необхідно отримати доступ. Биті адреси, зареєстровані як співпадаючі з командою READ або WRITE, використовуються для вибору місця розташування початкового стовпця для доступу до пакету. SDRAM забезпечує програмовані довжини пакетів читання (BL) або читання (записи), рівні 1, 2, 4 або 8 сторінкам з опція переривання читання Може бути включена функція автоматичної попередньої зарядки, щоб забезпечити самосинхронную попередню зарядку рядка, який ініціюється у кінці послідовності пакетів[25].

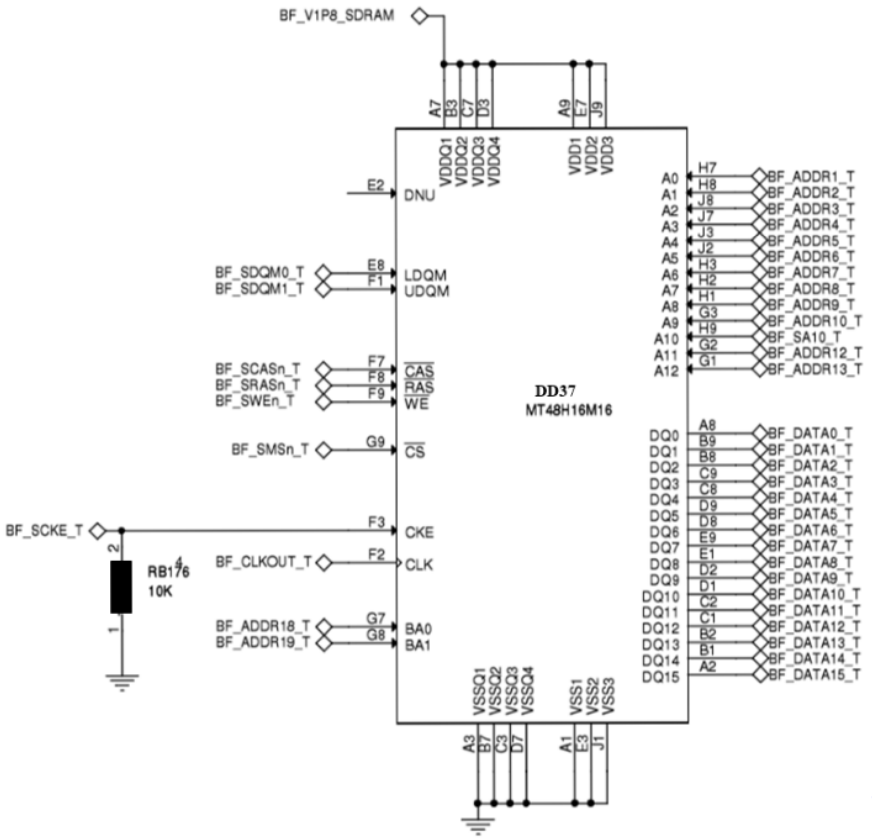


Рисунок 7.2- Принципова схема SDRAM пам’ять

256 МБ SDRAM використовує внутрішню конвеєрну архітектуру для досягнення високої швидкості роботи. Це також дозволяє змінювати адресу стовпця в кожному такті, щоб забезпечити високошвидкісний, повністю довільний доступ. Попередня зарядка одного банку при зверненні до одного з трьох інших банків дозволить приховати цикли попередньої зарядки і забезпечити безперебійну високошвидкісну операцію довільного доступу[25].

256-Мбайт SDRAM призначений для роботи в системах 1,8-вольтів з низьким енергоспоживанням. Передбачений режим автоматичного оновлення, а також режим глибокого відключення енергозбереження. Усі входи і виходи сумісні з LVTTL. SDRAM пропонує істотні поліпшення в продуктивності DRAM, включаючи можливість синхронної передачі даних з високою швидкістю передачі даних з автоматичною генерацією адреси стовпця, можливість чергування між внутрішніми банками, щоб приховати час попередньої зарядки і можливість

випадкової зміни адрес стовпців в кожному тактовому циклі під час пакетного доступу[25].

### 7.2.1 Функціональний опис

Як правило, 256 МБ SDRAM - це чотирьохканальна DRAM, яка працює при напрузі 1,8 В і включає синхронний інтерфейс (усі сигнали реєструються на позитивному фронті тактового сигналу, CLK). Доступ для читання і запису до SDRAM орієнтований на пакетну обробку; Доступ розпочинається з вибраного місця розташування і триває впродовж запрограмованої кількості місць в запрограмованій послідовності. Доступ розпочинається з реєстрації команди ACTIVE, за якою йде команда READ або WRITE. Зареєстровані біти адреси, співпадаючі з командою ACTIVE, використовуються для вибору банку і рядка, до якого здійснюється доступ (BA0 і BA1 вибирають банк, A0 - A12 вибирають рядок для x16, а A0 - A11 вибирають рядок для x32). Биті адреси (A0 - A8 для x16 і A0 - A8 для x32), зареєстровані одночасно з командою READ або WRITE, використовуються для вибору початкового місця розташування стовпця для пакетного доступу. Перед нормальною роботою SDRAM має бути ініціалізована[25].

## 7.3 Трансивер ADM1385

Наступний елемент траснивер ADM1385, принципова схема з фрагментом взаємодії трансивера, наведена на рисунку 7.3

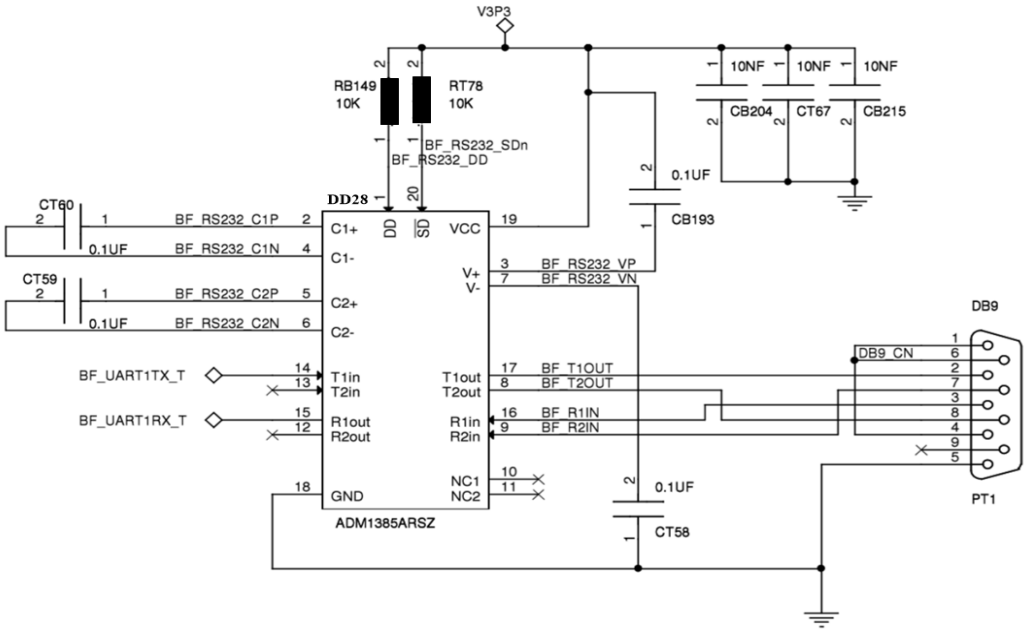


Рисунок 7.3.1 – Фрагмент принципової схеми з трансивером ADM1385

Трансивер ADM1385 - це високошвидкісне 2-канальне, інтерфейсне облаштування RS - 232 / V.28, працюючі від одного джерела живлення 3,3 В. Низьке енергоспоживання і можливість відключення роблять його ідеальними для портативних інструментів з батарейним живленням. Деталі ADM1385 відповідають специфікаціям EIA - 232e і CCITT V.28 і працюють на швидкостях передачі даних до 460 кбит / с. Чотири зовнішні конденсатори із зарядовою помпою 0,1 мкФ використовуються для подвоювача / інвертора напруги, що дозволяє працювати від одного джерела живлення 3,3. ADM1385 містить режим відключення драйвера і режим повного відключення[26].

Перетворювач напруги заряду складається з генератора з частотою 200 кГц і матриці перемикання. Перетворювач генерує живлення ± 6,6 В на вході рівня 3,3 В. Це робиться в два етапи з використанням методу перемиканих конденсаторів, як показано на рисунку 7.3.2 і рисунку 7.3.3. По-перше, вхідне живлення 3,3 В подвоюється до 6,6 В з використанням конденсатора C1 в якості елементу накопичення заряду. Рівень 6,6 В потім інвертується для генерування - 6,6 В з використанням C2 в якості елементу зберігання. C3 показаний підключеним між V і VCC, але однаково ефективний, якщо підключений між V і GND. Конденсатори

C3 і C4 використовуються для зменшення пульсацій на виході. Їх значення не є критичними і можуть бути збільшені за бажання. Конденсатор C3 показаний підключеним між V і VCC. Також допустимо підключати цей конденсатор між V і GND. За бажання конденсатори більшого розміру (до 10 мкФ) можна використати для конденсаторів C1 по C2[26].

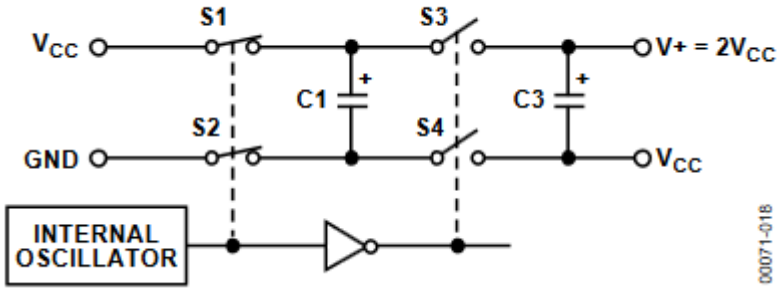


Рисунок 7.3.2

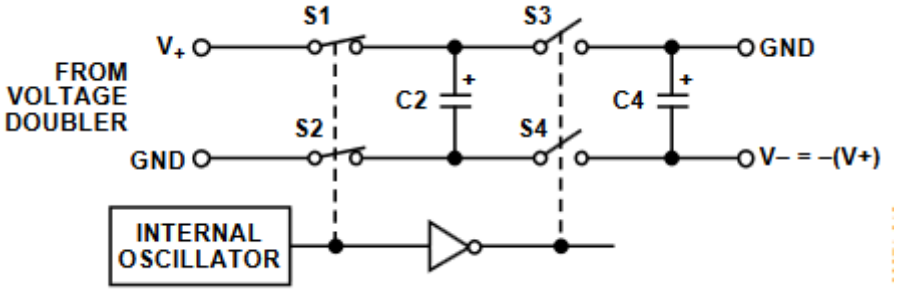


Рисунок 7.3.3

### 7.4 Флеш пам’ять N25Q

Наступний компонент флеш пам’ять моделі N25Q. N25Q - це високопродуктивне облаштування флеш пам’яті з декількома послідовними входами / виходами виготовлено за технологією NOR[27]. Він має функції виконання на місці (XIP), вдосконаленими механізмами захисту від запису і високошвидкісним SPI-сумісним інтерфейсом шини. Інноваційні, високопродуктивні та чотирьох ядерні інструкції введення / виведення дозволяють подвоїти пропускну спроможність передачі для операцій READ і PROGRAM. Пам'ять організована у вигляді 64 (64

КБ) основних секторів, які додатково розділені на 16 підсекторів кожен (всього 1024 підсектори). Пам'ять може бути видалена одним 4kb підсектором в годину, 64 КБ секторів за один раз або в цілому[27]. Пам'ять може бути захищена від запису програмним забезпеченням за допомогою енергозалежних і енергонезалежних функцій захисту, залежно від потреб додатка. Міра деталізації захисту 64kb (секторна гранулярна) для енергозалежних защит. Пристрій має 64 одноразових програмованих (OTP) байта, які можна прочитати і запрограмувати за допомогою команд READ OTP і PROGRAM OTP. Ці 64 байти також можуть бути назавжди заблокованні командою PROGRAM OTP. Пристрій також має можливість призупиняти і поновлювати цикли PROGRAM і ERASE, використовуючи спеціальні інструкції PROGRAM / ERASE SUSPEND і RESUME.

Пам'ять може працювати з трьома різними протоколами:

- Розширений SPI (стандартний протокол SPI, оновлений за допомогою операцій з двома і чотирма операціями)
- Подвійний вхід / вихід SPI
- Quad I / O SPI

Стандартний протокол SPI розширений за рахунок операцій з двома і чотирма операціями. Крім того, протоколи з двома і чотирма SPI скорочують час доступу до даних і пропускна спроможність одного облаштування введення-виведення при передачі команд, адрес і даних через дві або чотири рядки даних[27].

Висновок: Було розроблена принципова схема та розглянуто основні компоненти та елементи схеми.

## 8 Аналіз балістичного відхилення робочого елемента

Траєкторія кулі, при пострілі, залежить від дуже багатьох зовнішніх чинників. Розглянемо, деякі з них. Вітер вважається слабким при його швидкості 2 м/с, середнім (помірним), - при 4 м/с, сильним, - при 8 м/с. Бічний помірний вітер, що діє під кутом 90° до траєкторії, вже дуже значно впливає на легку кулю. Дія вітру

тієї ж сили, але що дме під гострим кутом до траєкторії - 45° і менше - викликає удвічі менше відхилення кулі. Вітер, що дме уздовж траєкторії в ту або іншу сторону, уповільнює або прискорює швидкість кулі, що треба враховувати при стрільбі по рухомій цілі[28].

Для кожної "далекобійної" зброї і боєприпасу існують свої таблиці поправок, що дозволяють враховувати вплив метеоумов і інших чинників на траєкторію польоту кулі. При оцінці результатів стрільби треба пам'ятати, що на кулю з моменту пострілу і до кінця її польоту діють деякі випадкові (що не враховуються) чинники, що призводить до невеликих відхилень траєкторії польоту кулі від пострілу до пострілу. Тому навіть в "ідеальних" умовах попадання кулі у ціль, траєкторія має вигляд овалу, що направляється до центру. Такі випадкові відхилення називаються девіацією. Розглянемо траєкторію польоту кулі та її елементи(Рис 8.1).

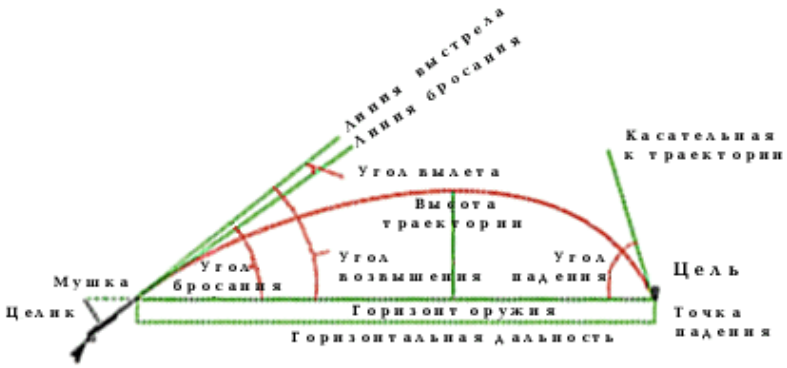


Рисунок 8.1 – Траєкторія польоту кулі.

Пряма лінія, що представляє продовження осі каналу ствола до пострілу, називається лінією пострілу. Пряма лінія, що є продовженням осі ствола при вильоті з нього кулі, називається лінією кидання. В результаті дії сили тяжіння і сили опору повітря, куля летить не по прямій траєкторії, а по нерівномірно розподіленій. Початком траєкторії є точка вильоту[28]. Горизонтальна площина, що проходить через точку вильоту, називається горизонтом зброї. Вертикальна площина, що проходить через точку вильоту по лінії кидання, називається

площиною стрільби. Кут, складений лінією пострілу і горизонтом зброї, називається кутом підвищення. Кут, складений лінією кидання і горизонтом зброї, називається кутом кидання. Точка перетину траєкторії з горизонтом зброї називається точкою падіння. Відстань по горизонту від точки вильоту до точки падіння називається горизонтальною дальністю. Кут між дотичною до траєкторії в точці падіння і горизонтом зброї називається (табличним) кутом падіння. Найвища точка траєкторії над горизонтом зброї називається вершиною траєкторії, а відстань від горизонту зброї до вершини траєкторії - заввишки траєкторії. Вершина траєкторії ділить траєкторію на дві нерівні частини[28].

### 8.1 Розрахунок похибок

Розрахуємо відносну похибку похибку для модуля далекоміра. Похибка вимірювання, яка заявлена виробником дорівнює -+0,5 м. За номінальну відстань візьмемо 1 км. Відносна похибка розраховується за формулою(8.2), де Δ=0,5м, а $X_{іст}$=1000м. Підставим відомі значення та розрахуємо відносну похибку(8.3).

$$\delta = \frac{\Delta}{X_{іст}} * 100\% \qquad (8.2)$$

$$\delta = \frac{0{,}5м}{1000м} * 100\% = 0{,}05\% \qquad (8.3)$$

Оскільки отримана відносна похибка рівномірно розподілена, то її межі позначені на рисунку 8.4,

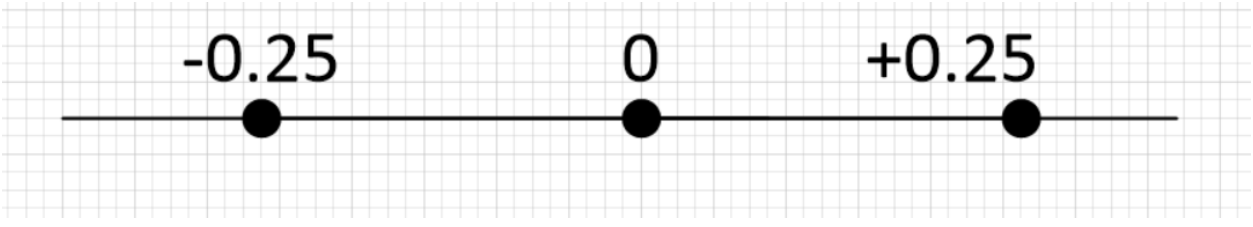


Рисунок 8.4

Розрахуємо СКВ випадкової похибки за формулою(8.5).

$$\sigma = \frac{\Delta}{\sqrt{12}} = \frac{0,5}{\sqrt{12}} = 0,144 \quad (8.5)$$

Ця похибка характеризує, як точно будуть відображені координати прицільної сітки. Також врахуємо, що при польоті куля буде схильна до впливу інших навколишніх факторів. Прирівняєм це відхилення до табличного значення у розмірі 1%. Звідси випливає, що систематична похибка дорівнює сумі СКВ випадкової похибки та 1%(8.6)

$$\delta_c = \sigma + 1\% = 0,144 + 1\% = 0,00144 \quad (8.6)$$

**8.2 Функція вимірювання**

Саме функція вимірювання буде визначати точність прицілу, тобто наскільки точно можна визначити зміщення прицілу. Наш відео дисплей має кругоподібну форму , тому за нульову точку координат ми беремо перетин вісей ординат. Оскілки це круг, то ми прирівнюємо кожну частину вісі до r, оскільки це радіус, який буде дорівнюватиме максимальному значенню зміщення об'єкта. Тоді, ми можемо прирівняти 1 піксель до 1 метру. Тобто, якщо приціл зміщається по вісі x наліво на 10 пікселей, це означає, що відбувається зміщення наліво на 10 метрів. Звідси випливає, що залежність r і значень балістичних таблиць і буде наша функція вимірювання. У результаті функція вимірювання буде носити табличну форму.  Функція вимірювання наведена нижче(8.7).

$$r_{vert} = \prod_{i,j,k,l}^{i\max, j\max, k\max, l\max} Select_{vert}[i, j, k, l]$$

$$r_{hor} = \prod_{i,j,k,l}^{i\max, j\max, k\max, l\max} Select_{hor}[i, j, k, l] \quad (8.7)$$

У даному випадку П це таблиця вертикального та горизонтального корегування. Параметри $i, j, k, l$ еквівалентні значенням температури, дистанції, вологості та тиску. Будь-яка балістична таблиця робиться під конкретні погодні умови. Адже температура, вологість, тиск і висота над рівнем моря теж мають значення. Це як правило погодні умови, в яких найчастіше працює стрілець. Але є таке поняття, як стандартні погодні умови. І якщо немає спеціальних погодних умов, то таблиця складається по стандартних: 59º F (температура); 78um (вологість); 29,53 inches (тиск 760 мм рт.стлб); 500 feet (висота)[9].

Висновок: Було розглянуто основні фактори впливу на траєкторію куліт а розраховані похибки вимірювання, такі як систематична, відносна та СКВ випадкової похибки. Також було побудовано функцію вимірювання для системи.

## 9 ОХОРОНА ПРАЦІ

Метою даного дипломного проекту є розробка системи тепловізійного прицілу. В процесі змішування відеопотоків вимірюються такі фізичні величини, як температура навколишнього середовища та відстань до цілі. Оскільки в процесі виконання технологічних операцій створення системи буде прибігатись використання пайки, пропоную розглянути основні правила при взаємодії з пайкою.

При проведенні робіт, пов'язаних з проведенням пайки і лудіння, необхідно чітко дотримуватися правил техніки безпеки. У зворотному випадку можна завдати шкоди своєму здоров'ю[29]. У роботі важливо використати якісні матеріали і інструменти. Припій пруток повинен відповідати усім стандартам. Припої використовують при пайці виробів з латуні, бронзи, міді. Керівники повинні провести грамотний інструктаж по роботі з цим інструментом.

Почати хотілося б з того, що до проведення робіт, пов'язаних з пайкою і лудінням, допускаються тільки особи, що досягли повноліття. Робітки повинні

пройти спеціальне навчання. Вони повинні досконало знати правила охорони праці, безпечні способи проведення робіт, уміти правильно поводитися з інструментами, пристосуваннями і вантажами. Якщо при виконанні пайки або лужки у працівника виникли які-небудь проблеми, він повинен звернутися до начальника, а не намагатися розв'язати проблему самостійно.

Украй серйозно треба відноситися до дотримання техніки безпеки, оскільки при пайці і лудінні, на працівника можуть впливати різні шкідливі чинники. До таких слід віднести підвищену загазованість повітря парами хімічних речовин, пожароопасность, бризки флюсів і припоїв, підвищену температуру повітря

Роботи, пов'язані з пайкою і лудінням, повинні проводитися в спеціально обладнаних і заздалегідь підготовлених приміщеннях. Обов'язково має бути присутньою система вентиляції. Вентиляційні установки мають бути оснащені звуковою і світловою сигналізацією.

У роботі важливо використати якісні і справні інструменти. Згідно з правилами технічної документації, паяльник повинен пройти спеціальну перевірку і випробування. Клас цього устаткування в обов'язковому порядку повинен відповідати умовам виробництва і категорії приміщення. Також треба потурбуватися про захист кабелю паяльника від зіткнення з гарячими предметами і захист від випадкового механічного ушкодження[29].

Висновок: Було розглянуто правила та основні моменти при взаємодії з пайкою, оскільки в процесі створення системи буде прибігатись використання пайки.

## Висновки

Результатом дипломного проєкту є система тепловізійного прицілу, яка значно розширює функціонал тепловізійного пристрою та виконує корегування координат прицілу за допомогою обробки параметрів навколишнього середовища. Дипломний проект повністю відповідає вимогами технічного завдання.

Протягом всього дипломного проєкту було спроектовано структурну, функціональну та принципову схеми пристроют. Також була розроблена програмне забезпечення для системи.

Було аналізовано та розглянуто основні фактори, які впливають на зміну траєкторії кулі та відповідно проведено аналіз похибок вимірювання.

Також було розглянуто розділ охорони праці, у якому були наведенні основні тези при роботі с пайкою, для успішного функціонування системи.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ


1. https://opticstrade.com/articles/vybiraem_teplovizionnyj_pricel
2. https://9mm.ru/catalog/product/teplovizor-pulsar-helion-xq28f/
3. https://9mm.ru/catalog/product/teplovizionnyy-pritsel-dedal-t2-380-venator/
4. https://tehpribory.ru/glavnaia/pribory/dalnomer.html
5. https://en.wikipedia.org/wiki/Laser_rangefinder
6. https://www.optics-pro.com.ua/novosti/stadiometricheskij-dalnomer-v-teplovizorah-i-pnv-kak-eto-rabotaet
7. https://aliexpress.ru/item/32802156306.html?spm=a2g0o.detail.1000013.2.64943fa9BX1guM&gpsid=pcDetailBottomMoreThisSeller&scm=1007.13339.146401.0&scm_id=1007.13339.146401.0&scm-url=1007.13339.146401.0&pvid=4ae9b038-bb64-46db-9e13-796dcd88d42a
8. https://aliexpress.ru/item/32860399159.html?spm=a2g0o.detail.1000013.8.7d5732f4AELswk&gps-id=pcDetailBottomMoreThisSeller&scm=1007.14977.161855.0&scm_id=1007.14977.161855.0&scm-url=1007.14977.161855.0&pvid=b06c93d5-9421-492b-84b5-e789e6f493bc&_t=gps-id:pcDetailBottomMoreThisSeller,scm-url:1007.14977.161855.0,pvid:b06c93d5-9421-492b-84b5-e789e6f493bc,tpp_buckets:668%230%23131923%2316_668%23808%234093%2374_668%23888%233325%237_668%232717%237564%23686
9. http://panhandleprecision.com/minimizing-trajectory-errors-know-conditions/
10. https://www.smarthof.ru/info/datchiki-temperatury/
11. http://mypractic.ru/ds18b20-datchik-temperatury-s-interfejsom-1-wire-opisanie-na-russkom-yazyke.html
12. http://www.shooting-ua.com/arm-books/arm_book_140.htm
13. http://electrik.info/main/automation/549-chto-takoe-mikrokontrollery-naznachenie-ustroystvo-princip-raboty-soft.html
14. STM32F217xx Data Sheet- Analog Devices, Inc., 2011, 168 с.

15. MicroCAM™ 3 Thermal Imaging Camera Manual- Thermoteknix Systems Ltd, 2016, 19 с .
16. https://hw-server.com/rs-232-overview-rs-232-standard
17. https://www.analog.com/media/en/technical-documentation/data-sheets/ADSP-BF522_BF523_BF524_BF525_BF526_BF527.pdf
18. Xcore LA series User Manual V1.5- © IRay Technology Co.,Ltd., 25 с.
19. Александр САМАРИН “Стандарты цифровых видеоинтерфейсов.”,2006, 9 с.
20. ADV7180 10-Bit SDTV Video Decoder Data Sheet- Analog Devices, Inc., 2015, 114 с.
21. Range Finder Module LW1012MU Data sheet Analog Devices, Inc.
22. NatureVue™ Video Signal Processor with Bitmap OSD, Dual HDMI Tx, and Encoder- Analog Devices, Inc.,2013, 64 с.
23. https://electronics.howstuffworks.com/hdmi.htm
24. DRV050-VGA-R02 Drive Board User manual- Yunnan North OLiGHTEK Opto-Electronic Technology Co., LTD, 2013, 8 с.
25. MT48H16M16LF Datasheet - Micron Technology, 2013, 71 с.
26. ADM3202/ADM3222/ADM1385 Data Sheet- Analog Devices, Inc., 2011,16с.
27. N25Q032A 32Mb, 3V Data sheet- Analog Devices, Inc., 2011, 81 с.
28. http://www.shooting-ua.com/force_shooting/practice_book_16.htm
29. http://www.znakcomplect.ru/poleznosti/example/texnika-bezopasnosti/texnika-bezopasnosti-pri-paike-i-luzhenii.html

# ДОДАТОК А

# ЛІСТИНГ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Main.cpp

**********/

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <setupapi.h>
#include "hostapp.h"
#include "usbdriver.h"
#include "console.h"

#include <vector>
#include <string>
using namespace std;

// guid for this device
EXTERN_C const GUID GUID_CLASS_BF_USB_BULKADI;

// SYNC/ASYNC defines, used to determine which method of I/O you want to use
#define USE_SYNC       FALSE
#define USE_ASYNC      TRUE

//////////////////////////////////////////////////
// prototypes
//////////////////////////////////////////////////
BOOL ConnectToDevice( BOOL bUseAsyncIo );
BOOL IsSupportedCommand( UINT uiCommand );
int Loopback( BOOL bRandomSize, UINT uiMaxDataBytes, UINT uiLoops = NULL
);
int DumpDeviceMemory( UINT uiAddress, UINT uiCount );
int FillDeviceMemory( UINT uiAddress, UINT uiCount, BYTE byValue );
int DownloadFile( CHAR *pszFilename, UINT uiAddress );
int UploadFile( CHAR *pszFilename, UINT uiAddress, UINT uiCount);
int GrabData();
int ConfigureADV(UINT script_number);
int ServeFileIoRequests();
int GetFirmwareVersionStrings(BYTE* pbyVersion);
int DisplayFirmwareVersionStrings();
int DoCustomCommand(vector<string> & vsCommand, string & strRetMsg);
void PrintUsage(bool shortForm = false);

//////////////////////////////////////////////////
```

```
// globals
//////////////////////////////////////////////////
DWORD g_dwTotalDevices = 0;                    // total devices detected
UINT g_unDeviceNumber = 0;                     // device number selected
char g_pszDeviceDesc[256];                     // device name string for dis-
play only
HANDLE g_hRead = INVALID_HANDLE_VALUE;         // USB device read han-
dle
HANDLE g_hWrite = INVALID_HANDLE_VALUE;        // USB device write
handle


/*****************************************************************************
*********
Routine Description:

Main function.

Arguments:

argc - argument count
argv - argument list

Return Value:

int - returns OPERATION_PASSED if successful else returns a failure code

*****************************************************************************
*********/
int _cdecl main(int argc, char *argv[])
{
int i;
char pszParam1[512];              // buffer storage for parameter 1
char pszParam2[512];              // buffer storage for parameter 2
char pszParam3[512];              // buffer storage for parameter 3
UINT uiParam1 = 0;                    // value of parameter 1
UINT uiParam2 = 0;                    // value of parameter 2
UINT uiParam3 = 0;                    // value of parameter 3
vector<string> vsCustomCommand;  // list of parameters for custom command
string strMsg;

int nRetVal = OPERATION_PASSED;       // return value

// save the current console window attributes
```

```
SaveConsoleAttributes();
SetRGB(WHITE);
printf("\n");

strcpy( g_pszDeviceDesc, "Blackfin USB" );        // device string

// make sure they entered a switch
if ( argc < 2 )
{
ColorPrintf(BR_RED, "\nPlease enter a valid switch, see usage below.\n" );
PrintUsage(TRUE);
}

for (i=0; i<argc && 1 < argc; i++)
{
if (argv[i][0] == '-')
{
// see how many devices are attached
g_dwTotalDevices = QueryNumDevices(
(LPGUID)&GUID_CLASS_BF_USB_BULKADI );

// get device number
g_unDeviceNumber = atoi(argv[i] + 2);

switch(argv[i][1])
{
// custom size loopback
case 'c':
case 'C':
// connect to the device
if (!ConnectToDevice( USE_ASYNC ))
return COULD_NOT_CONNECT;

// check to see if the firmware supports this command
if (!IsSupportedCommand( LOOPBACK ))
{
ColorPrintf(BR_RED, "\nThe current firmware does not support the -%c switch.\n",
argv[i][1]);
return UNSUPPORTED_COMMAND;
}

// if they didn't pass in a second parameter display usage
if (!(argv[i+1]))
{
```

```
ColorPrintf(BR_RED, "\nYou have not entered all the necessary parameters for this
command,\n");
ColorPrintf(BR_RED, "please see the usage below.\n");
PrintUsage(true);
break;
}

// else they passed it in
else
{
strcpy( pszParam1, argv[i+1] );                    // SIZE

// if they didn't pass in a third parameter we use the default for LOOPS
if (!(argv[i+2]))
strcpy( pszParam2, "0" );                  // LOOPS
else
strcpy( pszParam2, argv[i+2] );          // LOOPS

}

// see if it is in hex
if ( pszParam1[0] == '0' && (pszParam1[1] == 'x' || pszParam1[1] == 'X') )
sscanf( pszParam1, "%xd", &uiParam1 );
// else we assume decimal
else
sscanf( pszParam1, "%d", &uiParam1 );

// see if it is in hex
if ( pszParam2[0] == '0' && (pszParam2[1] == 'x' || pszParam2[1] == 'X') )
sscanf( pszParam2, "%xd", &uiParam2 );
// else we assume decimal
else
sscanf( pszParam2, "%d", &uiParam2 );

// see if they entered an invalid size
if ( (MAX_DATA_BYTES_BULKADI < uiParam1) || (uiParam1 <
MIN_DATA_BYTES_BULKADI) )
{
ColorPrintf(BR_RED, "You have entered an invalid size, please enter a value be-
tween\n");
ColorPrintf(BR_RED, "%d and ", MIN_DATA_BYTES_BULKADI);
ColorPrintf(BR_RED, "%d bytes.", MAX_DATA_BYTES_BULKADI);
break;
}
```

```
nRetVal = Loopback( FALSE, uiParam1, uiParam2 );
break;

// max size loopback
case 'l':
case 'L':
// connect to the device
if (!ConnectToDevice( USE_ASYNC ))
return COULD_NOT_CONNECT;

// check to see if the firmware supports this command
if (!IsSupportedCommand( LOOPBACK ))
{
ColorPrintf(BR_RED, "\nThe current firmware does not support the -%c switch.\n",
argv[i][1]);
return UNSUPPORTED_COMMAND;
}

nRetVal = Loopback( FALSE, MAX_DATA_BYTES_BULKADI );
break;

// random size loopback:
case 'r':
case 'R':
// connect to the device
if (!ConnectToDevice( USE_ASYNC ))
return COULD_NOT_CONNECT;

// check to see if the firmware supports this command
if (!IsSupportedCommand( LOOPBACK ))
{
ColorPrintf(BR_RED, "\nThe current firmware does not support the -%c switch.\n",
argv[i][1]);
return UNSUPPORTED_COMMAND;
}

nRetVal = Loopback( TRUE, MAX_DATA_BYTES_BULKADI );
break;

// detect devices
case 'a':
case 'A':
printf( "\nDetected %d ", g_dwTotalDevices);
```

```
ColorPrintf(BR_GREEN, "%s ", g_pszDeviceDesc);
if ( 1 == g_dwTotalDevices )
printf("device\n");
else
printf("devices\n");
break;

case 'b':
case 'B':
// connect to the device
if (!ConnectToDevice( USE_ASYNC ))
return COULD_NOT_CONNECT;

// check to see if the firmware supports this command
if (!IsSupportedCommand( CONFIGURE_ADV ))
{
ColorPrintf(BR_RED, "\nThe current firmware does not support the -%c switch.\n",
argv[i][1]);
return UNSUPPORTED_COMMAND;
}
if (!(argv[i+1]))
{
ColorPrintf(BR_RED, "\nYou have not entered all the necessary parameters for this
command,\n");
ColorPrintf(BR_RED, "please see the usage below.\n");
PrintUsage(true);
break;
}
else
{
strcpy( pszParam1, argv[i+1] );          // SCRIPT NUMBER
}

// see if it is in hex
if ( pszParam1[0] == '0' && (pszParam1[1] == 'x' || pszParam1[1] == 'X') )
sscanf( pszParam1, "%xd", &uiParam1 );
// else we assume decimal
else
sscanf( pszParam1, "%d", &uiParam1 );

nRetVal = ConfigureADV(uiParam1);
break;

// dump memory
```

```
case 'd':
case 'D':
// connect to the device
if (!ConnectToDevice( USE_ASYNC ))
return COULD_NOT_CONNECT;

// check to see if the firmware supports this command
if (!IsSupportedCommand( MEMORY_READ ))
{
ColorPrintf(BR_RED, "\nThe current firmware does not support the -%c switch.\n",
argv[i][1]);
return UNSUPPORTED_COMMAND;
}

// if they didn't pass in a second or third parameter display usage
if (!(argv[i+1]) || !(argv[i+2]) )
{
ColorPrintf(BR_RED, "\nYou have not entered all the necessary parameters for this
command,\n");
ColorPrintf(BR_RED, "please see the usage below.\n");
PrintUsage(true);
break;
}

// else they passed in a valid command line
else
{
strcpy( pszParam1, argv[i+1] );          // START
strcpy( pszParam2, argv[i+2] );          // COUNT
}

// see if it is in hex
if ( pszParam1[0] == '0' && (pszParam1[1] == 'x' || pszParam1[1] == 'X') )
sscanf( pszParam1, "%xd", &uiParam1 );
// else we assume decimal
else
sscanf( pszParam1, "%d", &uiParam1 );

// see if it is in hex
if ( pszParam2[0] == '0' && (pszParam2[1] == 'x' || pszParam2[1] == 'X') )
sscanf( pszParam2, "%xd", &uiParam2 );
// else we assume decimal
else
sscanf( pszParam2, "%d", &uiParam2 );
```

```
// see if they entered an invalid size
if ( (MAX_DATA_BYTES_BULKADI < uiParam2) || (uiParam2 <= 0) )
{
ColorPrintf(BR_RED, "\nYou have entered an invalid count, please enter a value be-
tween\n");
ColorPrintf(BR_RED, "1 and %d bytes.", MAX_DATA_BYTES_BULKADI);
break;
}

nRetVal = DumpDeviceMemory( uiParam1, uiParam2 );
break;

// download a file from the host to the device
case 'f':
case 'F':
// connect to the device
if (!ConnectToDevice( USE_ASYNC ))
return COULD_NOT_CONNECT;

// check to see if the firmware supports this command
if (!IsSupportedCommand( MEMORY_WRITE ))
{
ColorPrintf(BR_RED, "\nThe current firmware does not support the -%c switch.\n",
argv[i][1]);
return UNSUPPORTED_COMMAND;
}

// if they didn't pass in a second or third parameter display usage
if (!(argv[i+1]) || !(argv[i+2]) )
{
ColorPrintf(BR_RED, "\nYou have not entered all the necessary parameters for this
command,\n");
ColorPrintf(BR_RED, "please see the usage below.\n");
PrintUsage(true);
break;
}

// else they passed in a valid command line
else
{
strcpy( pszParam1, argv[i+1] );          // FILE
strcpy( pszParam2, argv[i+2] );          // START
}
```

```
// see if the address is in hex
if ( pszParam2[0] == '0' && (pszParam2[1] == 'x' || pszParam2[1] == 'X') )
sscanf( pszParam2, "%xd", &uiParam2 );
// else we assume decimal
else
sscanf( pszParam2, "%d", &uiParam2 );

nRetVal = DownloadFile( pszParam1, uiParam2 );
break;

case 'g':
case 'G':
// connect to the device
if (!ConnectToDevice( USE_ASYNC ))
return COULD_NOT_CONNECT;

// check to see if the firmware supports this command
if (!IsSupportedCommand( GRAB_DATA ))
{
ColorPrintf(BR_RED, "\nThe current firmware does not support the -%c switch.\n",
argv[i][1]);
return UNSUPPORTED_COMMAND;
}
nRetVal = GrabData();
break;

// download a file from device to the host
case 'i':
case 'I':
// connect to the device
if (!ConnectToDevice( USE_ASYNC ))
return COULD_NOT_CONNECT;

// check to see if the firmware supports this command
if (!IsSupportedCommand( MEMORY_READ ))
{
ColorPrintf(BR_RED, "\nThe current firmware does not support the -%c switch.\n",
argv[i][1]);
return UNSUPPORTED_COMMAND;
}

// if they didn't pass in a second or third or fourth parameter display usage
if (!(argv[i+1]) || !(argv[i+2]) || (!argv[i+3]))
```

```
{
ColorPrintf(BR_RED, "\nYou have not entered all the necessary parameters for this
command,\n");
ColorPrintf(BR_RED, "please see the usage below.\n");
PrintUsage(true);
break;
}

// else they passed in a valid command line
else
{
strcpy( pszParam1, argv[i+1] );          // FILE
strcpy( pszParam2, argv[i+2] );          // START
strcpy( pszParam3, argv[i+3] );          // COUNT
}

// see if the address is in hex
if ( pszParam2[0] == '0' && (pszParam2[1] == 'x' || pszParam2[1] == 'X') )
sscanf( pszParam2, "%xd", &uiParam2 );
// else we assume decimal
else
sscanf( pszParam2, "%d", &uiParam2 );

// see if the count is in hex
if ( pszParam3[0] == '0' && (pszParam3[1] == 'x' || pszParam3[1] == 'X') )
sscanf( pszParam3, "%xd", &uiParam3 );
// else we assume decimal
else
sscanf( pszParam3, "%d", &uiParam3 );

nRetVal = UploadFile( pszParam1, uiParam2, uiParam3 );
break;


// set memory
case 's':
case 'S':
// connect to the device
if (!ConnectToDevice( USE_ASYNC ))
return COULD_NOT_CONNECT;

// check to see if the firmware supports this command
if (!IsSupportedCommand( MEMORY_WRITE ))
{
```

```
ColorPrintf(BR_RED, "\nThe current firmware does not support the -%c switch.\n",
argv[i][1]);
return UNSUPPORTED_COMMAND;
}

// if they didn't pass in a second, third, or fourth parameter display usage
if (!(argv[i+1]) || !(argv[i+2]) || !(argv[i+3]) )
{
ColorPrintf(BR_RED, "\nYou have not entered all the necessary parameters for this
command,\n");
ColorPrintf(BR_RED, "please see the usage below.\n");
PrintUsage(true);
break;
}

// else they passed in a valid command line
else
{
strcpy( pszParam1, argv[i+1] );          // START
strcpy( pszParam2, argv[i+2] );          // COUNT
strcpy( pszParam3, argv[i+3] );          // VALUE
}

// see if it is in hex
if ( pszParam1[0] == '0' && (pszParam1[1] == 'x' || pszParam1[1] == 'X') )
sscanf( pszParam1, "%xd", &uiParam1 );
// else we assume decimal
else
sscanf( pszParam1, "%d", &uiParam1 );

// see if it is in hex
if ( pszParam2[0] == '0' && (pszParam2[1] == 'x' || pszParam2[1] == 'X') )
sscanf( pszParam2, "%xd", &uiParam2 );
// else we assume decimal
else
sscanf( pszParam2, "%d", &uiParam2 );

// see if it is in hex
if ( pszParam3[0] == '0' && (pszParam3[1] == 'x' || pszParam3[1] == 'X') )
sscanf( pszParam3, "%xd", &uiParam3 );
// else we assume decimal
else
sscanf( pszParam3, "%d", &uiParam3 );
```

```
// see if they entered an invalid size
if ( (MAX_DATA_BYTES_BULKADI < uiParam2) || (uiParam2 <= 0) )
{
ColorPrintf(BR_RED, "\nYou have entered an invalid count, please enter a value be-
tween\n");
ColorPrintf(BR_RED, "1 and %d bytes.", MAX_DATA_BYTES_BULKADI);
break;
}

// see if they entered an invalid value
if ( (0xff < uiParam3) || (uiParam3 <= 0) )
{
ColorPrintf(BR_RED, "\nYou have entered an invalid value, please enter a value be-
tween 0x00 and 0xff.");
break;
}

nRetVal = FillDeviceMemory( uiParam1, uiParam2, (BYTE)uiParam3 );
break;

// service file IO requests
case 'u':
case 'U':
// connect to the device
if (!ConnectToDevice( USE_SYNC ))
return COULD_NOT_CONNECT;

// check to see if the firmware supports this command
if (!IsSupportedCommand( USBIO_START ))
{
ColorPrintf(BR_RED, "\nThe current firmware does not support the -%c switch.\n",
argv[i][1]);
return UNSUPPORTED_COMMAND;
}

nRetVal = ServeFileIoRequests();
break;

// print usage
case 'h':
case 'H':
PrintUsage();
break;
```

```
// get firmware version info
case 'v':
case 'V':
// connect to the device
if (!ConnectToDevice( USE_ASYNC ))
return COULD_NOT_CONNECT;

// check to see if the firmware supports this command
if (!IsSupportedCommand( GET_FW_VERSION ))
{
ColorPrintf(BR_RED, "\nThe current firmware does not support the -%c switch.\n",
argv[i][1]);
return UNSUPPORTED_COMMAND;
}

nRetVal = DisplayFirmwareVersionStrings();
break;

// custom command, allows user to send down custom data to device
case 'x':
case 'X':

// connect to the device
if (!ConnectToDevice( USE_ASYNC ))
return COULD_NOT_CONNECT;

// check to see if the firmware supports this command
if (!IsSupportedCommand( CUSTOM_COMMAND ))
{
ColorPrintf(BR_RED, "\nThe current firmware does not support the -%c switch.\n",
argv[i][1]);
return UNSUPPORTED_COMMAND;
}

// save all parameters to command list first
for( i =2; i<argc; i++)
vsCustomCommand.push_back(argv[i]);

// execute custom command
nRetVal = DoCustomCommand(vsCustomCommand, strMsg);
if(!nRetVal)
{
// if device sends back a message, print it
if(!strMsg.empty())
```

```
printf("%s\n", strMsg.c_str());
// else if not just print generic success message
else
ColorPrintf(BR_GREEN, "Command successfully passed to device\n");
}
else
{
// custom command failed, print error message
ColorPrintf(BR_RED, "Command could not be passed to device\n");
}
break;

// unsupported switch
default:
ColorPrintf(BR_RED, "\n-%c is an unsupported switch, please see usage below.\n",
argv[i][1]);
PrintUsage(TRUE);
break;
}
}
}

// close handles
if( g_hRead && (g_hRead != INVALID_HANDLE_VALUE) )
CloseHandle( g_hRead );
if( g_hWrite && (g_hWrite != INVALID_HANDLE_VALUE) )
CloseHandle( g_hWrite );

printf("\n");

// restore the original console window attributes
RestoreConsoleAttributes();

// all done
return nRetVal;
}


/*****************************************************************************
*********
Routine Description:

Connect to a device based on the device number.
```

Arguments:

BOOL bUseAsyncIo - specifies if we should use ASYNC (TRUE) or SYNC (FALSE) IO

Return Value:

BOOL success (TRUE) or failure (FALSE)

**********/

```
BOOL ConnectToDevice( BOOL bUseAsyncIo )
{
// if no devices found print message and return
if ( !g_dwTotalDevices )
{
printf( "\nNo ");
ColorPrintf(BR_GREEN, "%s ", g_pszDeviceDesc);
printf("devices were found\n");
return FALSE;
}

// make sure it's a valid device number
if ( g_unDeviceNumber >= g_dwTotalDevices )
{
ColorPrintf(BR_RED, "Invalid device number %d\n", g_unDeviceNumber );
return FALSE;
}

// open write handle
g_hWrite = OpenDeviceHandle( (LPGUID)&GUID_CLASS_BF_USB_BULKADI,
WRITE_PIPE, g_unDeviceNumber, bUseAsyncIo );
if( g_hWrite == INVALID_HANDLE_VALUE)
{
ColorPrintf(BR_RED, "Error opening driver\n");
return FALSE;
}

// open read handle
g_hRead = OpenDeviceHandle( (LPGUID)&GUID_CLASS_BF_USB_BULKADI,
READ_PIPE, g_unDeviceNumber, bUseAsyncIo );
if( g_hRead == INVALID_HANDLE_VALUE)
{
ColorPrintf(BR_RED, "Error opening driver\n");
```

```
return FALSE;
}

// ok
return TRUE;
}

/*****************************************************************************
*********
Routine Description:

Query the firmware to see if it supports a given command.

Arguments:

UINT uiCommand - command we want to query for

Return Value:

BOOL success (TRUE) or failure (FALSE)

*****************************************************************************
*********/
BOOL IsSupportedCommand( UINT uiCommand )
{
USBCB usbcb;                                        // USB control
block
ULONG ulActualBytes = 0x0;                          // track actual bytes
ULONG ulCountRcvd = 0;                              // received byte
count
BOOL IoStatus = TRUE;                               // IO status


// send out a USBCB that tells the device we want to query for support
usbcb.u32_Command = QUERY_SUPPORT;                  // command
usbcb.u32_Count = 0x0;                              // doesn't matter
usbcb.u32_Data = uiCommand;                         // command to
query for

IoStatus = WritePipe(g_hWrite, &usbcb, sizeof(USBCB), &ulActualBytes);

// check for error
if (!IoStatus)
{
```

```
ColorPrintf(BR_RED, "\n\nError WRITING usbcb for QUERY FIRMWARE");
return FALSE;
}

// now we expect a USBCB back indicating the query result
ulCountRcvd = 0;
do
{
IoStatus = ReadPipe(g_hRead, (&usbcb) + ulCountRcvd, sizeof(usbcb), &ulActual-
Bytes);
ulCountRcvd += ulActualBytes;
} while ( ulCountRcvd < sizeof(usbcb) );

// check for error
if (!IoStatus)
{
ColorPrintf(BR_RED, "\n\nError READING usbcb for QUERY FIRMWARE");
return FALSE;
}

// u32_Data contains the result, either TRUE or FALSE
return (BOOL)usbcb.u32_Data;
}

/*****************************************************************************
*********
Routine Description:

Get the firmware version strings.

Arguments:

BYTE *pbyVersion -      pointer to a buffer to store the strings in

Return Value:

int -                   return status

*****************************************************************************
*********/
int GetFirmwareVersionStrings(BYTE *pbyVersion)
{
USBCB usbcb;                                        // command block
ULONG ulActualBytes;                                // track actual bytes
```

```
ULONG ulCountRcvd = 0;                                  // received byte count
BOOL IoStatus = TRUE;                                   // iostatus

// send command for firmware version
usbcb.u32_Command = GET_FW_VERSION;                             // command
usbcb.u32_Count = VERSION_STRING_BLOCK_SIZE;        // number of bytes
usbcb.u32_Data = 0x0;                                   // not used for this com-
mand

IoStatus = WritePipe(g_hWrite, &usbcb, sizeof(USBCB), &ulActualBytes);

// check for error
if (!IoStatus)
{
ColorPrintf(BR_RED, "\n\nError WRITING usbcb for FIRMWARE VERSION");
return IO_WRITE_USBCB_FAILED;
}

// now get the firmware version
ulCountRcvd = 0;
do
{
IoStatus = ReadPipe(g_hRead, pbyVersion + ulCountRcvd, VER-
SION_STRING_BLOCK_SIZE, &ulActualBytes);
ulCountRcvd += ulActualBytes;
} while ( ulCountRcvd < VERSION_STRING_BLOCK_SIZE );

// check for error
if (!IoStatus)
{
ColorPrintf(BR_RED, "\n\nError READING data for FIRMWARE VERSION");
return IO_READ_DATA_FAILED;
}

return OPERATION_PASSED;
}

/*****************************************************************************
*********
Routine Description:

Display the firmware version strings.

Arguments:
```

None

Return Value:

int - return status

**********/
```
int DisplayFirmwareVersionStrings()
{
BYTE pbyVersion[VERSION_STRING_BLOCK_SIZE];          // allocate
memory for the strings
int nRetVal = OPERATION_PASSED;                      // return value

nRetVal = GetFirmwareVersionStrings(pbyVersion);
if (OPERATION_PASSED == nRetVal)
{
// we are all set to display the strings
printf("\nFirmware version: ");
ColorPrintf(BR_GREEN, "%s", (char*)(pbyVersion + (FW_VERSION_NUM-
BER*MAX_VERSION_STRING_LEN)) );
printf("\nBuild date:       ");
ColorPrintf(BR_GREEN, "%s", (char*)(pbyVersion +
(FW_BUILD_DATE*MAX_VERSION_STRING_LEN)) );
printf("\nBuild time:       ");
ColorPrintf(BR_GREEN, "%s", (char*)(pbyVersion +
(FW_BUILD_TIME*MAX_VERSION_STRING_LEN)) );
printf("\nTarget processor: ");
ColorPrintf(BR_GREEN, "%s", (char*)(pbyVersion + (FW_TAR-
GET_PROC*MAX_VERSION_STRING_LEN)) );
printf("\nApplication name: ");
ColorPrintf(BR_GREEN, "%s", (char*)(pbyVersion + (FW_APPLICA-
TION_NAME*MAX_VERSION_STRING_LEN)) );
}

return nRetVal;
}
```

/**********
Routine Description:

Send custom command down to target board through USB as strings.  This allows

users to send generic string data down and interpret its meaning on the device side. Device must support CUSTOM_COMMAND. It uses the first argument as a sub-command which is coverted to a number, then sends any additional arguments as strings.

Arguments:

1: subcommand ID
2: string 0
3: ...

Return Value:

int - return status

**********/

```
int DoCustomCommand(vector<string> & vsCommand, string & strRetMsg)
{
int Result=0;
char *pcTemp;

// command must not be empty
if(vsCommand.empty())
{
// error, no sub-command found
return 0;
}

USBCB usbcb;                                        // command block
ULONG ulActualBytes;                                // track actual bytes
ULONG ulCountSent = 0;                              // sent byte count
ULONG ulCountRcvd = 0;                              // received byte count
BOOL IoStatus = TRUE;                               // iostatus

// send command for custom command
usbcb.u32_Command     = CUSTOM_COMMAND;             // command
usbcb.u32_Count       = vsCommand.size()-1;  // next data length for sub-command

// convert sub-command to number
if(!vsCommand[0].empty())
usbcb.u32_Data        = atoi(vsCommand[0].c_str());

//send number of strings down to DSP via USBCB
```

```
IoStatus = WritePipe(g_hWrite, &usbcb, sizeof(USBCB), &ulActualBytes);

// check for error
if (!IoStatus)
{
ColorPrintf(BR_RED, "\n\nError WRITING usbcb for CUSTOM COMMAND");
return FALSE;
}

// send down each string to DSP
for(unsigned int i =1; i< vsCommand.size(); i++)
{
// length of string
usbcb.u32_Count          = vsCommand[i].length();          // next data length for
sub-command

// send USBCB with length
IoStatus = WritePipe(g_hWrite, &usbcb, sizeof(USBCB), &ulActualBytes);

ulActualBytes = 0;
pcTemp = const_cast<char*>(vsCommand[i].data());
int len = vsCommand[i].length();

do
{
// send the data
IoStatus = WritePipe(g_hWrite, pcTemp, vsCommand[i].length(), &ulActualBytes);
if (!IoStatus)
{
break;
}
ulCountSent += ulActualBytes;
pcTemp += sizeof(char)*ulActualBytes;
} while ( ulCountSent < vsCommand[i].length() );

// check for error
if (!IoStatus)
{
ColorPrintf(BR_RED, "\n\nError WRITTING buffer for CUSTOM COMMAND");
return FALSE;
}
}

// read back from device which tells us if we are receiving a result string
```

```
ulCountRcvd = 0;
usbcb.u32_Count  =0;
do
{
IoStatus = ReadPipe(g_hRead, &usbcb, sizeof(USBCB), &ulActualBytes);
ulCountRcvd += ulActualBytes;
} while ( ulCountRcvd < sizeof(USBCB) );

// check for error
if (!IoStatus)
{
ColorPrintf(BR_RED, "\n\nError READING usbcb for CUSTOM COMMAND");
return FALSE;
}

if(usbcb.u32_Count)
{
pcTemp = new char[usbcb.u32_Count+1];

//we have more data to read
ulCountRcvd = 0;
do
{
IoStatus = ReadPipe(g_hRead, pcTemp, usbcb.u32_Count, &ulActualBytes);
ulCountRcvd += ulActualBytes;
} while ( ulCountRcvd < usbcb.u32_Count );

// check for error
if (!IoStatus)
{
ColorPrintf(BR_RED, "\n\nError READING buffer for CUSTOM COMMAND");
return FALSE;
}
pcTemp[usbcb.u32_Count] = '\0';
strRetMsg = pcTemp;
delete [] pcTemp;
}
return OPERATION_PASSED;
}


/*****************************************************************************
*********
Routine Description:
```

Display usage.

Arguments:

bool shortForm - flag to indicate if we should print example command lines

Return Value:

None

```
**********************************************************************
*********/
void PrintUsage(bool shortForm)
{
// print command line with all possible switches
ColorPrintf(BR_WHITE, "\n\nhostapp ");
ColorPrintf(RED, "-a");
printf(" | ");
ColorPrintf(YELLOW, "-c SIZE LOOPS");
printf(" | ");
ColorPrintf(GREEN, "-d START COUNT");
printf(" | ");
ColorPrintf(CYAN, "-f FILE START");
printf(" | ");
ColorPrintf(MAGENTA, "-h");
printf(" | ");
ColorPrintf(BR_RED, "-l");
printf(" | ");
ColorPrintf(BR_YELLOW, "-r");
printf(" | ");

printf("\n       ");                    // newline

ColorPrintf(BR_GREEN, "-s START COUNT VALUE");
printf(" | ");
ColorPrintf(BR_CYAN, "-u");
printf(" | ");
ColorPrintf(BR_MAGENTA, "-v\n");

// explain all the switches
ColorPrintf(RED,      "\n -a                ");printf("- display auto-detected devices and
their device numbers");
```

```
ColorPrintf(MAGENTA,   "\n -b SCRIPT_NUM        ");printf("- configures ADV part with particular script number");
ColorPrintf(YELLOW,    "\n -c SIZE LOOPS        ");printf("- run custom loopback with SIZE byte transfers LOOP times");
ColorPrintf(GREEN,     "\n -d START COUNT       ");printf("- dump COUNT bytes of memory at START address");
ColorPrintf(CYAN,      "\n -f FILE START        ");printf("- download FILE from host to device at START address");
ColorPrintf(MAGENTA,   "\n -g                   ");printf("- sets Blackfin for grabbing data");
ColorPrintf(MAGENTA,   "\n -h                   ");printf("- display usage");
ColorPrintf(BR_RED,    "\n -i FILE START COUNT  ");printf("- dumps the COUNT-bytes from the device to FILE at START address");
ColorPrintf(BR_RED,    "\n -l                   ");printf("- run maximum size loopback test");
ColorPrintf(BR_YELLOW, "\n -r                   ");printf("- run random size loopback test");
ColorPrintf(BR_GREEN,  "\n -s START COUNT VALUE ");printf("- set COUNT bytes of memory at START address with VALUE");
ColorPrintf(BR_CYAN,   "\n -u                   ");printf("- service IO requests from de-vice");
ColorPrintf(BR_MAGENTA,"\n -v                   ");printf("- display firmware version in-formation");

ColorPrintf(BR_WHITE,"\n\nNote:");
printf("  If multiple devices are connected you can add a device number to the");
printf("\nswitch in order to access a particular device, otherwise the default is 0.");

// print out sample command lines
if (!shortForm)
{
ColorPrintf(BR_WHITE,"\n\nExample usage:\n");
ColorPrintf(BR_WHITE,"\nhostapp ");
ColorPrintf(BR_RED,  "-l");                         printf("          - run loopback on de-vice 0");
ColorPrintf(BR_WHITE,"\nhostapp ");
ColorPrintf(YELLOW,  "-c2 512 1000");    printf("  - run 512 byte loopback for 1000 loops on device 2");
ColorPrintf(BR_WHITE,"\nhostapp ");
ColorPrintf(GREEN,   "-d3 0x0 0x100");   printf(" - dump 0x100 bytes from address 0x0 on device 3");
ColorPrintf(BR_WHITE,"\nhostapp ");
ColorPrintf(BR_CYAN, "-u");                         printf("          - service IO requests from device 0");
```

```
}
}

/*****************************************************************************
*********
Routine Description:

Sends command "GRAB_DATA" to Blackfin initializing PPI and DMA

Arguments:

None

Return Value:

int - return status

*****************************************************************************
*********/

int GrabData( )
{
USBCB       usbcb;                                                  // USB
command block
BOOL        IoStatus = TRUE;                                // iostatus
int             nRetVal = OPERATION_PASSED;                     // return value
ULONG ulActualBytes = 0x0;                                  // track actual bytes

// send packet telling we want to do a MEMORY_WRITE
usbcb.u32_Command = GRAB_DATA;                  // command
usbcb.u32_Count = (ULONG)0;                     // number of bytes
usbcb.u32_Data = (ULONG)0;          // start address

IoStatus = WritePipe(g_hWrite, &usbcb, sizeof(USBCB), &ulActualBytes);

// check for error
if (!IoStatus)
{
ColorPrintf(BR_RED, "\n\nError sending GRAB command");
nRetVal = IO_WRITE_USBCB_FAILED;
}

return nRetVal;
}
```

```
/*****************************************************************************
*********
Routine Description:

Sends command "CONFIGURE_ADV"

Arguments:

UINT script_number - script number

Return Value:

int - return status

*****************************************************************************
*********/

int ConfigureADV(UINT script_number)
{
USBCB       usbcb;                                                  // USB
command block
BOOL        IoStatus = TRUE;                                // iostatus
int         nRetVal = OPERATION_PASSED;                     // return value
ULONG   ulActualBytes = 0x0;                                // track actual bytes

// send packet telling we want to do a MEMORY_WRITE
usbcb.u32_Command = CONFIGURE_ADV;                  // command
usbcb.u32_Count = (ULONG)0;                                 // number of bytes
usbcb.u32_Data = (ULONG)script_number;              // script number

IoStatus = WritePipe(g_hWrite, &usbcb, sizeof(USBCB), &ulActualBytes);

// check for error
if (!IoStatus)
{
ColorPrintf(BR_RED, "\n\nError sending CONFIGURE_ADV command");
nRetVal = IO_WRITE_USBCB_FAILED;
}

return nRetVal;
}
```

**Memory.cpp**

```
**********************************************************************
**********/

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <setupapi.h>
#include "hostapp.h"
#include "usbdriver.h"
#include "console.h"

////////////////////////////////////////////////
// extern globals
////////////////////////////////////////////////

extern DWORD g_dwTotalDevices;          // total devices detected
extern UINT g_unDeviceNumber;           // device number selected
extern char g_pszDeviceDesc[];          // device name string for display only
extern HANDLE g_hRead;                          // USB device read handle
extern HANDLE g_hWrite;                         // USB device write handle

/**********************************************************************
**********
Routine Description:

Dumps Blackfin memory

Arguments:

UINT uiAddress - Indicates if we should loop random data sizes
UINT uiCount -          If random this indicates the maximum transfer size, else
indicates the transfer size of each loop

Return Value:

int - return status

**********************************************************************
**********/

int DumpDeviceMemory( UINT uiAddress, UINT uiCount )
{
char *pcInBuff = NULL;                  // input buffer
char *pcOutBuff = NULL;                         // output buffer
```

```
char pszAddress[64];                    // address buffer
char *pcTemp;                                   // temp pointer
USBCB *pusbcb = NULL;                   // USB command block pointer
ULONG nCountRcvd = 0;                   // count received
ULONG nCountCurrent = 0;                // current count
UINT uiRemainder = 0;                   // remainder
ULONG i = 0, j = 0, k = 0;              // indexes
ULONG nActualBytes;                     // actual bytes transferred
BOOL IoStatus = TRUE;                   // IO status

// allocate some storage for buffer
pcOutBuff = new char [sizeof(USBCB)];
pcInBuff = new char [uiCount];

// point usbcb to front of new buffer, point temp pointer to input buffer
pusbcb = (USBCB*)pcOutBuff;
pcTemp = pcInBuff;

// send packet telling we want to do a MEMORY_READ
pusbcb->u32_Command = MEMORY_READ;              // command
pusbcb->u32_Count = uiCount;                    // number of bytes
pusbcb->u32_Data = (ULONG)uiAddress;  // start address

IoStatus = WritePipe(g_hWrite, pusbcb, sizeof(USBCB), &nActualBytes);

// check for error
if (!IoStatus)
{
ColorPrintf(BR_RED, "\n\nError WRITING usbcb for MEMORY DUMP");

// free the buffers
if ( pcOutBuff )
delete [] pcOutBuff;
if ( pcInBuff )
delete [] pcInBuff;
return IO_WRITE_USBCB_FAILED;
}

do
{
IoStatus = ReadPipe(g_hRead, pcTemp, sizeof(char) * (pusbcb->u32_Count),
&nCountCurrent);

// check for error
```

```
if (!IoStatus)
{
ColorPrintf(BR_RED, "\n\nError READING data for MEMORY DUMP");
break;
}

// add to the total we've already received
nCountRcvd += nCountCurrent;

// advance pointer on host
pcTemp += sizeof(char) * nCountCurrent;

} while ( nCountRcvd < uiCount );

// check for error
if (!IoStatus)
{
ColorPrintf(BR_RED, "\n\nError READING data for MEMORY DUMP");

// free the buffers
if ( pcOutBuff )
delete [] pcOutBuff;
if ( pcInBuff )
delete [] pcInBuff;
return IO_WRITE_DATA_FAILED;
}

// now print the buffer
for ( i = 0; i < uiCount; i++ )
{
// every 8 bytes
if ( !(i % 8) )
{
// after printing a line, print the last 8 ASCII chars
if (i)
{
printf (" ");

// loop through each character
SetRGB(MAGENTA);
for ( k = 8; k > 0; k-- )
{
// if it is not a printable character print a space instead
if ( (0x20 <= pcInBuff[i - k]) && (0x7e >= pcInBuff[i - k]) )
```

```
{
printf ("%c", pcInBuff[i-k]);
}
else
printf (" ");
}
SetRGB(WHITE);
}

// print a newline and new address
sprintf( pszAddress, "%8.8X", uiAddress + i );
SetRGB(CYAN);
printf("\n0x%s  ", pszAddress);
SetRGB(WHITE);
}

// print the hex value
SetRGB(YELLOW);
printf("%2.2X ", (BYTE)(pcInBuff[i]));
}
SetRGB(WHITE);

// we should have leftover ASCII chars to print
uiRemainder = uiCount % 8;
if ( !uiRemainder )
uiRemainder = 8;

// need to line it up, account for missing hex values
for (j = 0; j < (8 - uiRemainder)*3 + 1; j++)
printf(" ");

for (j = 0; j < uiRemainder; j++)
{
// if it is not a printable character print a space instead
if ( (0x20 <= pcInBuff[i - uiRemainder + j]) && (0x7e >= pcInBuff[i - uiRemainder +
j]) )
ColorPrintf(MAGENTA, "%c", pcInBuff[i - uiRemainder + j]);
else
printf(" ");
}

// free the buffers
if ( pcOutBuff )
delete [] pcOutBuff;
```

```
if ( pcInBuff )
delete [] pcInBuff;

return OPERATION_PASSED;
}


/*****************************************************************************
*********
Routine Description:

Fills Blackfin memory

Arguments:

UINT uiAddress - Indicates if we should loop random data sizes
UINT uiCount -          If random this indicates the maximum transfer size, else
indicates the transfer size of each loop
BYTE byValue -          Value to fill memory with

Return Value:

int - return status

*****************************************************************************
*********/

int FillDeviceMemory( UINT uiAddress, UINT uiCount, BYTE byValue )
{
char *pcBuffer;                                 // buffer to store data
char *pcTemp;                                   // temp pointer
UINT i;                                                 // index
ULONG nCountSent = 0;                           // count sent
ULONG nCountCurrent = 0;                // current count sent
USBCB usbcb;                                    // USB command block
BOOL IoStatus = TRUE;                           // IO status

// allocate and initialize a buffer of data
pcBuffer = new char [uiCount];
if ( !pcBuffer )
{
ColorPrintf(BR_RED, "\n\nUnable to allocate memory for this operation");
return OUT_OF_MEMORY_ON_HOST;
}
```

```
for ( i = 0; i < uiCount; i++ )
pcBuffer[i] = byValue;

// send packet telling we want to do a MEMORY_WRITE
usbcb.u32_Command = MEMORY_WRITE;                    // command
usbcb.u32_Count = (ULONG)uiCount;                    // number of bytes
usbcb.u32_Data = (ULONG)uiAddress;                   // start address

pcTemp = pcBuffer;

IoStatus = WritePipe(g_hWrite, &usbcb, sizeof(USBCB), &nCountCurrent);

// check for error
if (!IoStatus)
{
ColorPrintf(BR_RED, "\n\nError WRITING usbcb for MEMORY SET");
if ( pcBuffer )
delete [] pcBuffer;
return IO_WRITE_USBCB_FAILED;
}

do
{
IoStatus = WritePipe(g_hWrite, pcTemp, sizeof(char) * (usbcb.u32_Count -
nCountSent), &nCountCurrent);

// check for error
if (!IoStatus)
{
break;
}

// add to the total we've already sent
nCountSent += nCountCurrent;

// advance pointer on host
pcTemp += sizeof(char) * nCountCurrent;

} while ( nCountSent < uiCount );

// check for error
if (!IoStatus)
{
ColorPrintf(BR_RED, "\n\nError WRITING data for MEMORY SET");
```

```
if ( pcBuffer )
delete [] pcBuffer;
return IO_WRITE_DATA_FAILED;
}

ColorPrintf(BR_WHITE, "Memory has been set.");

if ( pcBuffer )
delete [] pcBuffer;

return OPERATION_PASSED;
}


/*****************************************************************************
*********
Routine Description:

Downloads a host file to the Blackfin

Arguments:

CHAR *pszFilename -    File on host we want to download.
UINT uiAddress - Address file is downloaded to.

Return Value:

int - return status

*****************************************************************************
*********/

int DownloadFile( CHAR *pszFilename, UINT uiAddress )
{
FILE  *InFile;                                          // input file
char   pcBuffer[MAX_DATA_BYTES_BULKADI];          // buffer to store file
data
int            nActual = 0;                             // count read in from
file
int            nTotal = 0;                                      // total count read
in from file
USBCB      usbcb;                                                  // USB
command block
ULONG      ulCurrentAddr = uiAddress;                   // address pointer
```

```
ULONG     nActualBytes = 0;                    // actual bytes passed to
the driver
BOOL      IoStatus = TRUE;                     // iostatus
int       nRetVal = OPERATION_PASSED;                  // return value


// open input file as binary
if( (InFile = fopen( pszFilename, "rb" )) == NULL )
ColorPrintf(BR_RED, "The input file was not found, please check the path and try
again.\n" );

// check for error opening the file
if (!InFile)
{
ColorPrintf(BR_RED, "\nError opening input file, please make sure the file is accessi-
ble and try again.");
return ERROR_OPENING_FILE;
}

// loop until the end of file is found
while( !feof( InFile ) )
{
// attempt to read in MAX_DATA_BYTES_BULKADI bytes
nActual = fread( pcBuffer, sizeof( char ), MAX_DATA_BYTES_BULKADI, InFile );
if( ferror( InFile ) )
{
ColorPrintf(BR_RED, "\nError reading input file, download will not complete." );
nRetVal = ERROR_WRITING_FILE;
break;
}

// send packet telling we want to do a MEMORY_WRITE
usbcb.u32_Command = MEMORY_WRITE;                  // command
usbcb.u32_Count = (ULONG)nActual;                  // number of bytes
usbcb.u32_Data = (ULONG)ulCurrentAddr;             // start address

IoStatus = WritePipe(g_hWrite, &usbcb, sizeof(USBCB), &nActualBytes);

// check for error
if (!IoStatus)
{
ColorPrintf(BR_RED, "\n\nError WRITING usbcb for FILE DOWNLOAD");
nRetVal = IO_WRITE_USBCB_FAILED;
break;
```

```
}

IoStatus = WritePipe(g_hWrite, pcBuffer, sizeof(char) * usbcb.u32_Count, &nActual-
Bytes);

// check for error
if (!IoStatus)
{
ColorPrintf(BR_RED, "\n\nError WRITING data for FILE DOWNLOAD");
nRetVal = IO_WRITE_DATA_FAILED;
break;
}

// increment counters
nTotal += nActual;
ulCurrentAddr += nActual;
}

ColorPrintf(BR_WHITE, " \nFile \"%s\" has been downloaded.", pszFilename );

// close file
if (InFile)
{
if( fclose( InFile ) )
ColorPrintf(BR_RED, "\nThe input file could not be closed." );
}

return nRetVal;
}

/*
**************************************************************************
************************** */
/*
**************************************************************************
************************** */
/*
**************************************************************************
************************** */
/*
**************************************************************************
************************** */

int ReadMemory( UINT uiAddress, UINT uiCount, char *pcInBuff)
```

```
{
char *pcOutBuff = NULL;                 // output buffer
char pszAddress[64];                    // address buffer
char *pcTemp;                           // temp pointer
USBCB *pusbcb = NULL;                   // USB command block pointer
ULONG nCountRcvd = 0;                   // count received
ULONG nCountCurrent = 0;                // current count
UINT uiRemainder = 0;                   // remainder
ULONG i = 0, j = 0, k = 0;              // indexes
ULONG nActualBytes;                     // actual bytes transferred
BOOL IoStatus = TRUE;                   // IO status

// allocate some storage for buffer
pcOutBuff = new char [sizeof(USBCB)];

// point usbcb to front of new buffer, point temp pointer to input buffer
pusbcb = (USBCB*)pcOutBuff;
pcTemp = pcInBuff;

// send packet telling we want to do a MEMORY_READ
pusbcb->u32_Command = MEMORY_READ;          // command
pusbcb->u32_Count = uiCount;                // number of bytes
pusbcb->u32_Data = (ULONG)uiAddress;  // start address

IoStatus = WritePipe(g_hWrite, pusbcb, sizeof(USBCB), &nActualBytes);

// check for error
if (!IoStatus)
{
ColorPrintf(BR_RED, "\n\nError WRITING usbcb for MEMORY DUMP");

// free the buffers
if ( pcInBuff )
delete [] pcInBuff;
return IO_WRITE_USBCB_FAILED;
}

do
{
IoStatus = ReadPipe(g_hRead, pcTemp, sizeof(char) * (pusbcb->u32_Count),
&nCountCurrent);

// check for error
if (!IoStatus)
```

```
{
ColorPrintf(BR_RED, "\n\nError READING data for MEMORY DUMP");
break;
}

// add to the total we've already received
nCountRcvd += nCountCurrent;

// advance pointer on host
pcTemp += sizeof(char) * nCountCurrent;

} while ( nCountRcvd < uiCount );

// check for error
if (!IoStatus)
{
ColorPrintf(BR_RED, "\n\nError READING data from Blackfin");

// free the buffers
if ( pcOutBuff )
delete [] pcOutBuff;
return IO_WRITE_DATA_FAILED;
}

// free the buffers
if ( pcOutBuff )
delete [] pcOutBuff;

return OPERATION_PASSED;
}


/*****************************************************************************
*********
Routine Description:

Downloads a host file to the Blackfin

Arguments:

CHAR *pszFilename -    File on host we want to download.
UINT uiAddress - Address file is downloaded to.

Return Value:
```

```
int - return status

**********************************************************************
**********/
int UploadFile( CHAR *pszFilename, UINT uiAddress, UINT uiCount)
{
FILE  *OutFile;                                                 // output file
UINT    uiReadNow;
UINT    Result = OPERATION_PASSED;
UINT    uiCountTemp;
char    *pcInBuff;
char    *pcInTemp;

pcInBuff = new char [uiCount];
pcInTemp = pcInBuff;
uiCountTemp = uiCount;

while (uiCountTemp)
{
uiReadNow = min(uiCountTemp, MAX_DATA_BYTES_BULKADI);
printf("\n - Readmemory: %08X, %08X", uiAddress, uiReadNow);
if (ReadMemory( uiAddress, uiReadNow, pcInTemp) != OPERATION_PASSED)
{
ColorPrintf(BR_RED, "\n\nError READING data for MEMORY DUMP");
break;
}
pcInTemp += uiReadNow; // move pointer
uiAddress += uiReadNow;
uiCountTemp -= uiReadNow;
}

// open input file as binary
if( (OutFile = fopen( pszFilename, "wb" )) == NULL )
ColorPrintf(BR_RED, "The output file was not created, please check the path and try
again.\n" );

// check for error opening the file
if (!OutFile)
{
ColorPrintf(BR_RED, "\nError opening output file.");
return ERROR_OPENING_FILE;
}
```

```
if (fwrite( pcInBuff, sizeof( char ), uiCount,  OutFile) != uiCount)
{
ColorPrintf(BR_RED, "\nError writing output file. (disk full?)" );
Result = ERROR_WRITING_FILE;
}

// release memory
if ( pcInBuff )
delete [] pcInBuff;

// close file
if( fclose( OutFile ) )
ColorPrintf(BR_RED, "\nThe output file could not be closed." );

return Result;
}
```

**ДОДАТОК Б**

**СХЕМА ЕЛЕКТРИЧНА ПРИНЦИПОВА**

**ПЕРЕЛІК ЕЛЕМЕНТІВ**